Firefox 16 Megabyte Memory Module Functional Specification

Revision 3.0

Bill Bruce (MEMORY::BRUCE)
John Pare (MEMORY::PARE)
Steve Rochefort (MEMORY::ROCHEFORT)

ESD
333 South Street
Shrewsbury, MA 01545

December 31, 1987

RESTRICTED DISTRIBUTION

Blank Page

Table of Contents

Blank Page

Revision History

Date	Version	Changes
18 Nov 87	4.0	Added MBRQ in Mem Writes and arb error checking
04 Nov 87		Added command code for READU transaction
20 Jul 87		Incorporated changes requested by M. Nielsen
13 Jul 87		Fixed SBE Syndrome (Hex) error for D<53:50>
01 May 87	3.0	Changed bus protocol to include MDATINV, MBUSY Deleted interlock circiutry Added BUSERR, BUSCTL, BUSDAT registers, and revised FMDCSR register
07 Apr 87	2.0	Add signals section, new power section Revised and expanded register section Revised module self test and cooperative test strategy section
06 Feb 87	1.0	Integrate with System Specification
04 Feb 87	0.0	Preliminary Document Creation

Blank Page

9. Firefox 16 Megabyte Memory Module

9.1. Scope

This specification is intended to provide all the details necessary to utilize a Firefox 16M Byte Memory Module as a primary memory in a M-bus system application. The functions of the memory module will be described from both the hardware and software viewpoint.

9.1.1. Overview

The Firefox 16M Byte Memory Module is a primary memory which responds to the M-bus protocol and retains/retrieves data to be used by the Firefox processors and I/O subsystem. This module's storage capacity is 16 megabytes of ECC protected DRAM memory which may be accessed via the M-bus protocol specified in the "Firefox M-Bus Specification." This module performs all the translations necessary to interface the M-bus to the DRAMs from a timing and protocol standpoint during normal operation. This module DOES NOT have battery backup capability and all data will be lost in the event of a power failure.

This module has the capability of performing self test and initialization of its DRAM array on command from a processor. Also, most of the control and interface circuitry (M-bus and DRAM) can be verified by exercising the module with a sequence of M-bus commands and checking the residue of a Linear Feedback Shift Register via an I/O Read cycle. The object of this testing is to provide the system user with a high degree of confidence in the memory subsystem.

Figure 9-1 shows the block diagram of this memory module.

9.1.2. Related Documents

The following documents contain material that is referenced:

- Firefox System Specification
- Firefox M-Bus Specification
- Firefox Bus Interface Chip Specification
- Dynamic MOS RAM, +5V, 1M (1048576 × 1), Fast Page Mode Specification

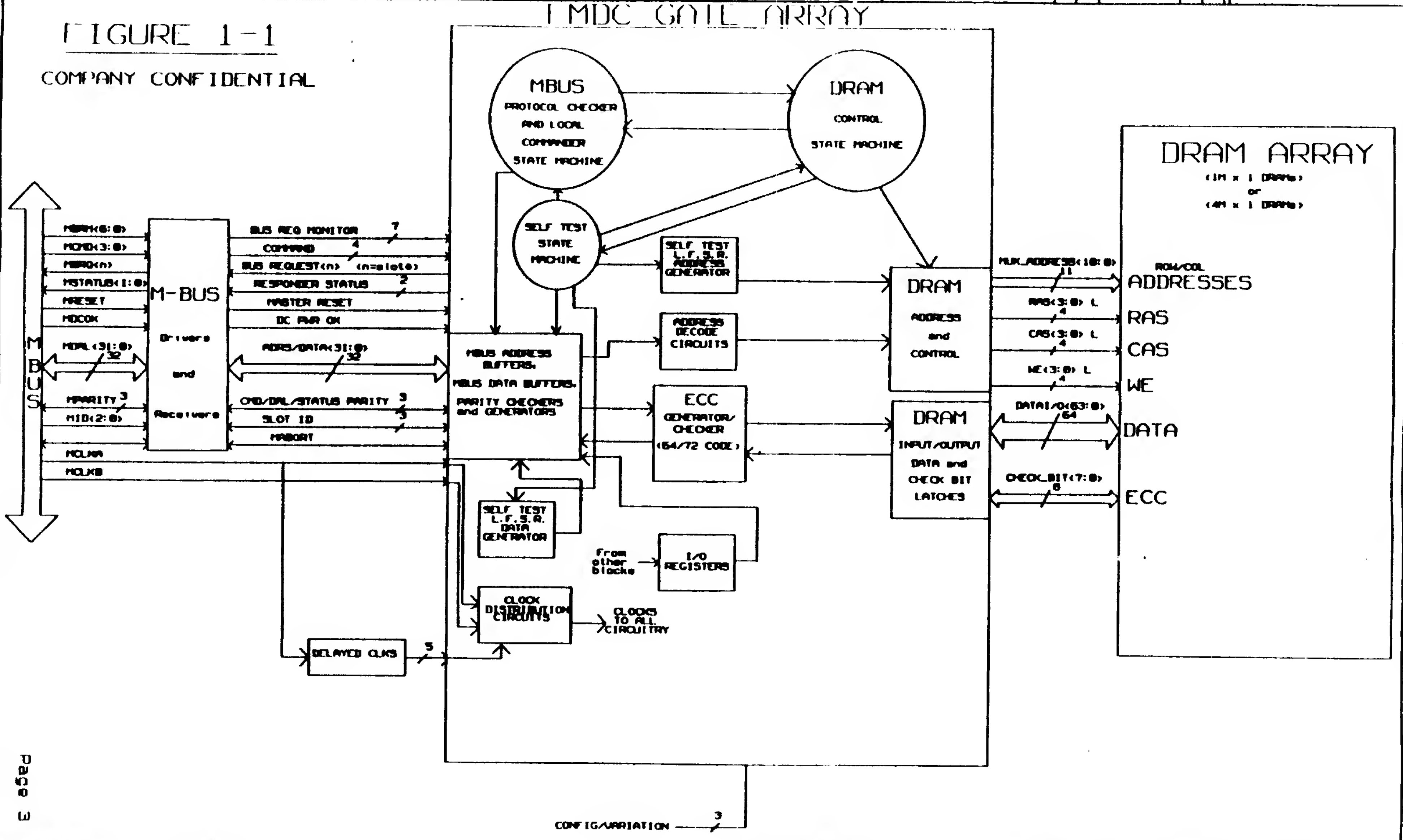
9.1.3. Terminology

CAS	Column Address Strobe. This term is used to describe a control signal sent to a DRAM. One function of this signal is to indicate that the DRAM should interpret the signals on its address pins as the second dimension of its two dimensional addressing scheme and latch them internally. Another function is to serve as an "output enable" which controls the driving of the DRAM's Data Out pin during a READ operation. This signal, with RAS and WE, controls the operation of the DRAM.
Column Address or Column	This term is used to describe the second half of the two dimensional, time multiplexed addressing scheme used to access data in most DRAMs. The addresses are present on the DRAM address input pins and their interpretation is controlled by CAS. The first half of the address was previously presented as the Row Address.
DBE	Double Bit Error. This term is describes an indication given by the ECC circuitry that two data bits of the data you have been checking are in error. This class of error is not correctable with the ECC code used in this memory system.
DRAM	Dynamic Random Access Memory. A form of Read/Write memory whose contents must be periodically refreshed so that data will be maintained.

ECC	Error Correcting Code. In the Firefox memory system, the ECC circuitry implements a modified Hamming code operating on 64 data bits internal to the memory module and using 8 check bits (64/72 code). This code is designed to correct all single bit errors and detect all double bit errors.
FMDC	Firefox Memory Data path and Control gate array. This term is used to describe the semicustom gate array which implements most of the functions of the memory system (except actual data storage and M-bus drivers/tranceivers).
MBE	Multiple Bit Error. This term is describes an indication given by the ECC circuitry that two, or more, data bits of the data you have been checking are in error.
MBZ	Must Be Zero. This term is used to indicate that a field of an I/O register must be driven as all "0's". This mechanism is used to reserve register bits for future enhancements and make new versions of a product backward compatible.
RAS	Row Address Strobe. This term is used to describe a control signal sent to a DRAM. One function of this signal is to indicate that the DRAM should interpret the signals on its address pins as one dimension of its two dimensional addressing scheme and latch them internally. Another function is to serve as an flag to start an internal set of timing sequences that lead to a memory access. This signal, with CAS and WE, controls the operation of the DRAM.
Refresh	This term is used to describe an operation that must be performed periodically on a DRAM so that it will retain the data stored in it. In the Firefox memory system, the FMDC performs this function and makes it nearly invisible to the M-bus.
Row Address or Row	This term is used to describe the first half of the two dimensional, time multiplexed addressing scheme used to access data in most DRAMs. The addresses are present on the DRAM address input pins and are latched by RAS. The second half of the address is presented as the Column Address.
SBE	Single Bit Error. This term is describes an indication given by the ECC circuitry that one data bit of the data you have been checking is in error. This class of error is correctable with the ECC code used in this memory system.
SMT	Surface Mount Technology. This term is used in this document to describe any integrated circuit packaging technology that is not the standard "through hole" method (ie. SOJ and SOIC packages).
SOIC	Small Outline Integrated Circuit. This term is used to describe an industry standard surface mounted packaging technology which uses a much smaller physical package than the standard DIP(Dual Inline Package) and has a "gull wing" style lead bend.
SOJ	Small Outline "J" Leaded IC. This term is used to describe an industry standard surface mounted packaging technology which uses a much smaller physical package than the standard DIP(Dual Inline Package) and has a "J" style lead bend.
WE	Write Enable. This signal, when asserted and in conjunction with RAS and CAS, signals the DRAM to store data from its Data IN pin into its internal array.

FIGURE 1-1

COMPANY CONFIDENTIAL



Page 3

digital	Bill Bruce	1-30-87	TITLE: FIREFOX MEMORY Gate Array/Module BLOCK DIAGRAM	
			SIZE D	CODE BD	REV A

9.2. M-Bus Receivers/Transceivers/Drivers

Table 9-1 lists the transceiver/driver components that are utilized on the Memory M-Bus interface. All tranceiver/driver components are in SMT packages (SOICs). Specified termination will be connected in series immediately after the bus driver transceiver output (bus side of transceivers). All specified pullups will be implemented on the backplane.

Table 9-1: M-Bus Transceiver/Driver Components

Component	Termination	Pullup	Signal(s)
74F244	20 ohm	4.7K ohm	MBRQ
74F245	20 ohm	4.7K ohm	MDAL<31:24>
74F245	20 ohm	4.7K ohm	MDAL<23:16>
74F245	20 ohm	4.7K ohm	MDAL<15:08>
74F245	20 ohm	4.7K ohm	MDAL<07:00>
74F245	20 ohm	4.7K ohm	MDPARITY
74F245	20 ohm	4.7K ohm	MCMD<3:0>, MCPARITY
74F245	20 ohm	4.7K ohm	MSTATUS<1:0>, MSPARITY
74AS760		143/768 ohm	MDATINV, MBUSY, MABORT

Table 9-2 lists the per memory module loading for each M-bus signal. Loading is in addition to the output driver, if appropriate. For example, the MABORT signal has 1 74AS760 output driver and 1 CMOS input load per memory array module.

Table 9-2: Memory Array Module input Loading

Loading	Signal(s)
1 CMOS INPUTS	MBUSY, MDATINV, MABORT, MRESET, MDCOK
1 74F245 TRANSCEIVER	MDAL, MDPARITY
1 74F245 TRANSCEIVER	MCMD, MCPARITY
1 74F245 TRANSCEIVER	MSTATUS, MSPARITY
1 CMOS INPUTS	MCLKA
1 TTL (-4ma) INPUT	MCLKA
1 CMOS INPUTS	MCLKB

9.2.1. Signals

Since the memory interfaces directly to the M-bus, the signal list and functions are defined in the "Firefox M-Bus Specification." The signal descriptions contained herein are as they relate to the memory specifically.

9.2.1.1. Bus Arbitration

In contrast to the operation of the CPU, the memory does not, strictly speaking, arbitrate for the M-bus. The reasons for this are:

a) The memory is never a bus master and therefore does not need to arb for mastership.

b) During shared transactions, the memory knows one cycle ahead not to assert its MBRQ.

c) There should never be a case for more than one memory to be holding the same locations except in the case of a slot MID failure or memory resource allocation error. The latter case would signal a MABORT during P4 or P7 because of a multiple MBRQ error.

For the above reasons, the memory is not required to check priority level of other requestors but only that there is one and only one requestor on the bus during non-arb cycles.

9.2.1.1.1. MBRQ

The memory drives one of the eight MBRQ signals to show that it is the one that is responding to an M-bus transaction. The MBRQ that is driven is a function of the slot in which the module is inserted.

The module in slot #0 will drive MBRQ<0> etc.

The memory module which is selected by the transaction address, either I/O address or memory space address, asserts its MBRQ signal for as long as it drives the bus. In the case of shared memory space transactions, the memory will forfeit, to the highest priority cache, the right to drive the bus.

Although the memory does not arbitrate for the bus, it must monitor all seven of the other MBRQ signals. The reason for this is:

a) It must determine if it is the bus slave

b) It must keep in synchronization with the rest of the system

c) It must check for multiple MBRQs during a non-arbitration cycle.

The memory will assert its MBRQ during the time that the system is asserting MRESET. The system will use this feature to determine which slots are occupied. The MBRQ assertion and deassertion will be one cycle delayed from MRESET because of the pipelined nature of the bus protocol.

9.2.1.2. Data Transfer

Data transfer to and from the memory is accomplished via the 32 bit bidirectional MDAL bus under control of the MCMD and MSTATUS signals. The MPARITY signals enable single bit error detection of the MCMD, MSTATUS, and MDAL signals.

The MDAL signals are driven by the bus master to specify address, and write data. The MDAL signals are driven by the memory to specify read data. The MCMD signals are driven by bus masters to specify the transaction type and I/O space byte masks. The MSTATUS signals are driven by the memory to indicate status of the current transaction.

9.2.1.2.1. MCMD

The MCMD lines, which are driven by the bus master, serve two different purposes during various transaction times: transaction type during P2 of all bus transactions; and I/O read/write byte mask during P3 of I/O space transactions.

The following table lists the encoding of the MCMD lines during cycle P2 of a bus transaction:

Table 9-3: MCMD Line During Cycle P2

Value	Mnemonic	Function
0101	READ	I/O read or mem space read request
0111	WRITET	Mem space write through request
1001	READI	Interlocked mem space read request
1010	WRITE	I/O write or mem space write request
1011	WRITEU	Mem space write unlock request
*1101	INTACK	Interrupt acknowledge request
1110	READU	I/O read or mem space read unshared request

* The INTACK transaction is not utilized by the memory.

Note that a memory space versus an I/O space transaction is specified by MDAL<31> of a read or write address.

For an I/O space read or write transaction, the MCMD lines specify byte mask for the longword being accessed. MCMD<3:0> correspond to longword byte <3:0>. If MCMD<n> is asserted then the corresponding byte of the longword contains valid data. If MCMD<n> is deasserted, then the corresponding byte of the longword contains undefined data. Although some of the bytes in a longword may be undefined, all 32 lines plus the parity line must be driven. The parity will be calculated always across the entire longword.

When the master detects internal errors on data being written to memory, it asserts the MDATINV signal corresponding to the bad long word. By specifying "data is invalid" via MDATINV, the memory can force an uncorrectable ECC code for the quadword.

9.2.1.2.2. MSTATUS

The MSTATUS lines are asserted by the selected memory during I/O transactions and unshared memory space read transactions. The memory must actively assert WAIT status to stall a transaction until it is ready with read status. The following table describes the values and functions of the MSTATUS lines.

Table 9-4: Values and Functions of the MSTATUS Lines

Value	Mnemonic	Function
00	WAIT	Stall transaction
01	GOOD	Write complete/good read data
10	CORRECTED	Corrected read data after SBE
11	ERROR	(this status unused by memory)

9.2.1.2.3. MDAL

The MDAL are bidirectional lines which are used by the memory to specify I/O address, memory space address, read data or write data. They also indicate interrupt level and vector for modules which perform interrupt acknowledge transactions. Note that although the memory does not perform interrupt transactions, it does check parity on the MDAL during the INTACK.

9.2.1.2.4. MPARITY

The MCPARITY signal specifies even parity for the MCMD signals. The MSPARITY specifies even parity for the MSTATUS signals. The MDPARITY signal specifies even parity for the MDAL signals. When the bus master is driving the MCMD signals or the MDAL signals the memory will check those fields for correct parity. When the memory is driving the MSTATUS or the MDAL signals it will drive the entire field and the corresponding parity. See Section 9.4.5.1 for a detailed explanation of what and when to check parity.

9.2.1.2.5. MSHARED

The MSHARED signal is used by the caches to maintain data integrity throughout the system. The memory does not utilize this signal but a pin will be reserved on the FMDC in case of future changes.

9.2.1.2.6. MDATINV

Whenever a module drives data onto the MDAL that is known to contain errors, it asserts MDATINV. When the memory receives a longword of data accompanied by the MDATINV signal, it will force bad ECC for that quadword. If the memory supplies read data which had an uncorrectable ECC code, it will assert MDATINV during the two longwords corresponding to the quadword in error.

9.2.1.3. Control Signals

The MID, MRESET, MDCOK, MIRQ, and MABORT signals are used to initialize and coordinate activity on the M-bus.

9.2.1.3.1. MID

The MID signals uniquely identify each backplane slot with a value from 0 to 7. The table below lists the connections for each slot. The MID value is used by the address decode logic in the memory to determine what I/O addresses the memory will respond to. In this way, no jumpers or switches will be required for configuring a module.

Table 9-5: MID Signal Connections for Backplane Slots

Slot	MID<2>	MID<1>	MID<0>
0	Gnd	Gnd	Gnd
1	Gnd	Gnd	+5V
2	Gnd	+5V	Gnd
3	Gnd	+5V	+5V
4	+5V	Gnd	Gnd
5	+5V	Gnd	+5V
6	+5V	+5V	Gnd
7	+5V	+5V	+5V

9.2.1.3.2. MRESET

The MRESET signal will be used at bus initialization time to set the memory to a known state. When the MRESET signal is asserted, the memory will terminate any current transaction. On the asserted to deasserted transition of MRESET, the refresh function will be initiated and the memory address space will be made inaccessible. The memory address space of each memory module will remain inaccessible until:

a) The base address register has been loaded to specify the start address of a module's memory address space and the MEMSPEN bit in the BASEADDR register has been asserted.

b) Though not absolutely necessary, we recommend that the DRAM self-test function be performed so that all locations are initialized with proper ECC throughout the memory address space (see self test section).

9.2.1.3.3. MDCOK

Since the memory is not equipped with a battery backup feature, it will not respond or be required to react in any way to the MDCOK signal.

9.2.1.3.4. MIRQ

Since the memory does not utilize the interrupt function, it will never assert or be required to react in any way to the MIRQ signals. There will be no pin reserved on the FMDC for these signals.

9.2.1.3.5. MABORT

The MABORT signal will be asserted by the memory if it detects an error condition during any M-bus transaction. The possible transaction errors are:

a) bus arbitration errors (multiple masters or slaves)

b) parity errors on MDAL, MCMD, or MSTATUS

c) reserved values on the MCMD during P2

d) too many slave WAIT or MBUSY cycles

For a more detailed description of the MABORT function, see the section on error strategy.

9.2.1.4. Clocks

The MCLKA and MCLKB master clocks are used by the memory to control all of its timing. The MCLKI signal functions as an interval timer for the system and is not utilized by the memory.

9.2.1.4.1. MCLKA

MCLKA is the master clock for the M-bus. All signal transitions and bus interface state machines are referenced to MCLKA. The M-bus cycles, Pn, are defined by the rising edge of MCLKA, i.e., MCLKA is used by the memory to clock registers and enable latches driving the M-bus. MCLKA is radially distributed to each module in the backplane in order to minimize skew between modules.

9.2.1.4.2. MCLKB

MCLKB is the slave clock for the M-bus. M-bus receiver latches and registers in the memory will be clocked by MCLKB. MCLKB is radially distributed to each module slot in order to minimize skew between modules.

9.2.1.4.3. M-Bus MCLKA/MCLKB Waveforms

Figure 9-1 shows the waveforms expected by the memory subsystem. The clock generator is based on a divide-by-6 circuit of the master oscillator. The clock generator is free-running and self-initializing from an arbitrary power-up state within four oscillator cycles. The memory receives only the MCLKA and MCLKB signals. The OSC trace is included only to show the six phase nature of the clocking system. The memory worst case design will assume a minimum cycle period of 65ns (MCLKA rising to MCLKA rising). The maximum clock cycle period will be 80ns. The memory will operate at cycle periods greater than 80ns but the refresh period of the DRAMs will rise to a value greater than specifications allow. As long as the occurrence of DRAM data loss is taken into account, the memory may be operated at cycle periods greater than 80ns.

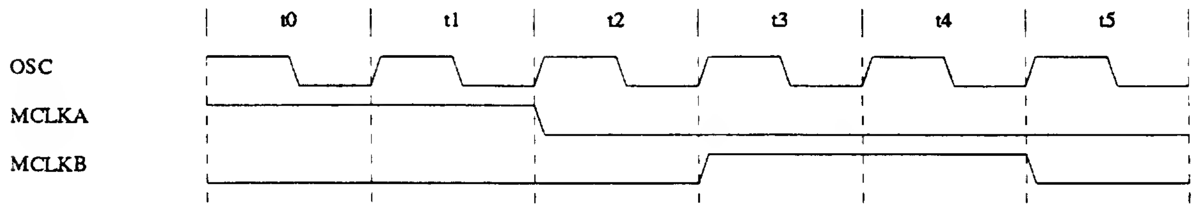


Figure 9-1: Waveforms Expected by the Memory Subsystem

9.3. DRAM Array

The memory module's gate array will designed to allow for up to three different memory types using 1M × 1 DRAMs and up to three different memory types using 4M × 1 DRAMs (six in total). The chart below outlines the six different memory type possibilities and the memory capacity associated with each one:

Table 9-6: Memory Types and Capacity

Array of Type	No. Memory DRAMs	DRAM Type Capacity (Bytes)	1M x 1	4M x 1
8 Mbyte	72	8,388,608 + ECC	X	
16 Mbyte	144	16,777,216 + ECC	X	
*32 Mbyte	288	33,544,432 + ECC	X	
*32 Mbyte	72	33,544,432 + ECC		X
*64 Mbyte	144	67,108,864 + ECC		X
*128 Mbyte	288	134,217,728 + ECC		X

* Not currently planned

One can see from the chart each array type has either 72, 144, or 288 DRAMs. Correspondingly, each memory type contains either one, two, or four banks of DRAMs. For example, modules which contain 144 DRAMs contain two banks of memory chips no matter what size DRAM is used. Bank control will be performed by additional RAS signals.

ECC (8 bits) will be generated by the array on two longwords (64 bits). Therefore, the 16 megabyte array can be viewed as two banks of DRAMs, 72 bits wide and 1,048,576 bits deep.

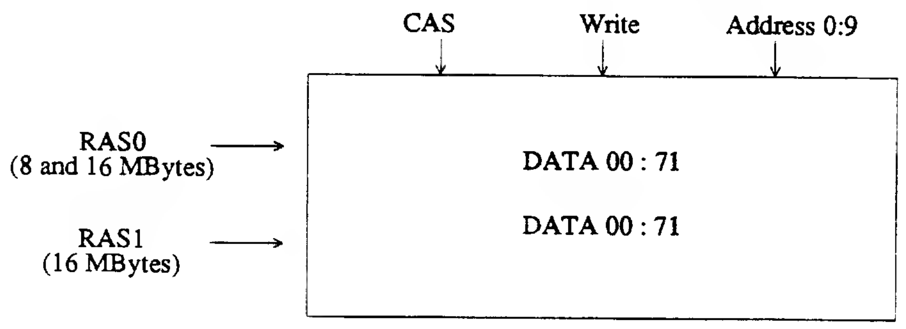


Figure 9-2: DRAM Array Organization in 8 and 16 MBytes Modules

All memory chips used in our 8 and 16 Mbyte designs will be fast page mode 1 M × 1 DRAMs (26 pin SOJ package). These arrays will be single -sided surface mount technology boards.

74F series drivers and series damping resistors will be be used to buffer the RAS, CAS, WRITE, and ADDRESS lines. Four octal driver packages will be used on the 8 Mbyte module and eight packages will be used on the 16 Mbyte array. This will afford the following loading on the DRAM control signals:

Table 9-7: Loading on DRAM Control Signals

Control Signal	DRAM Loading
ADDRESS<10:0>	36
RAS	18
CAS	18
WRITE	36

9.4. FMDC (Firefox Memory Data path and Control) Gate Array

The following sections describe the characteristics of this primary element in the Firefox memory system.

9.4.1. Address Path

The memory module's addressing consists of row and column DRAM addresses as well as DRAM bank select. Address decode has been designed to recognize up to six different array types to allow for flexibility in future memory expansion. The array types and address signal assignments are listed below:

Module Types:

8, 16, or 32 Mbyte Module using 1M x 1 DRAMs

32, 64, or 128 MByte Module using 4M x 1 DRAMs

Address Signals Assignments:

MDAL <13:04> = Row Addresses (all array sizes)

MDAL <19:14> = Column Addresses (all array sizes)

MDAL <22:20> = Column Addresses (all array sizes normalized using BASEADDR)

MDAL <23> =
> 8 Mbyte module Offset Address line *. Bank Select for 16 Mbyte and 32 Mbyte modules which use 1M x 1 DRAMs. Row Address for all 4M x 1 DRAM based modules.

MDAL <24>=
> Offset Address line* for 8 and 16 Mbyte modules. Bank Select for 32 Mbyte module which uses 1M x 1 DRAMs. Column Address for all 4M x 1 DRAM based modules.

MDAL <25>=
> Offset Address line* for 8, 16, and 32 Mbyte modules. Bank Select for 64 and 128 Mbyte modules

MDAL <26>=
> Offset Address line* for 8, 16, 32, and 64 Mbyte modules. Bank Select for 128 Mbyte module which uses 4M x 1 DRAMs.

MDAL <30:27> = Offset Address lines* for all memory modules

MDAL <31> = Indicates I/O Space Address transaction

9.4.2. Data Path

The Data Path section of the Memory Array module perfoms several basic functions.

- Carries write data from the M-bus to the DRAMS
- Carries read data from the DRAMS to the M-bus
- Stores a line of write data, in the case of a refresh in progress, until the DRAMS are ready

* - "Offset Address lines" are used by the array, in conjunction with the BASEADDR register, to determine whether or not to respond to the current M-bus transaction. DRAM addresses in this range are normalized to zero before they are used.

- Maintains data integrity between the M-bus and the DRAMS
- Reports to the M-bus interface in the event of a correctable or uncorrectable error

9.4.2.1. Write Data Buffer

As a result of a WRITE or WRITEU, write data is loaded from the MDAL lines into the write data path. Parity is checked on each 32 bit "long word" as it is received from the bus. The MDATINV signal is observed to determine if corrupted data is knowingly being presented by the bus master. The data is then assembled into quadwords and is presented to the ECC circuits one quadword at a time. The ECC circuits create an eight bit code which is combined with the write data to form a 72 bit data word. The two 72 bit words are assembled, one behind the other, so that they may be transferred to 72 dynamic RAMS during a double page mode cycle. The two 72 bit words may be held in the data path for a specified time in the event of an in progress RAM refresh cycle.

9.4.2.2. Read Data Buffer

As a result of a READ or READI, an octaword of read data is taken from 72 dynamic RAMS by means of a double page mode read cycle. The two 72 bit words are taken, one at a time, and presented to the ECC circuits to check for correctable or un-correctable errors. In the event of a correctable error, data will be corrected and SBE status will accompany the data toward the M-bus. In the event of an uncorrectable error, data will be left uncorrected and "MBE" status will accompany the data toward the M-bus. Each of the quadwords will then be divided into 32 bit "long words" and parity will be generated. To complete the READ transaction, the four 33 bit words (data + parity) will be sent to the M-bus along with "corrected" status or "good" status to reflect a SBE or no error respectively. If an MBE is encountered, the MDATINV signal will be asserted along with the data.

9.4.2.3. ECC

The ECC circuits implement a Hamming code by which all single bit errors can be corrected and multiple bit or address parity errors can be detected. During WRITE cycles, eight "check bits" are generated based on a 64 bit quadword and two bits of address parity. This eight bit code plus the quadword are stored in 72 DRAMS. During READ cycles, the 72 bits are read and ECC is again generated on the quadword + address parity. The newly generated ECC is compared with the check bits just read from RAM to determine a syndrome. The syndrome codes can then be used to determine which bit will be automatically corrected or what type of un-correctable error was detected.

9.4.3. Memory M-Bus Interface

Since the memory array interfaces directly to the Firefox M-bus, (it doesn't utilize a private memory bus) it must respond to M-bus transactions and adhere to the M-bus specification and protocol. The memory M-bus interface does differ from some of the other M-bus modules in that it never takes the role of bus master. This implies that the memory can only respond to bus transactions or monitor the bus. The purpose of monitoring the M-bus in the case of "un-owned" transactions is to stay in synchronization with bus transactions and to aid in checking bus protocol and bus information integrity.

The Memory M-bus interface responds to the following transaction types:

- READ (memory space read or I/O read)
- READI (read interlocked)
- READU (read unshared)
- WRITE (memory space write or I/O write)
- WRITET (write through)
- WRITEU (write unlock)

The memory will never initiate a INTACK (interrupt acknowledge) command. It will monitor an INTACK in order to check bus protocol and bus parity.

9.4.3.1. Memory Space Read (unshared)

During P2 of an M-bus transaction, with MDAL bit 31 unasserted, the memory will interpret a READ command as a memory space read. The memory M-bus interface will then decode the upper address bits to determine, based on the base address reg and the memory size, if the address falls within its domain. If the address is "owned," and the MEMSPEN bit (BASEADDR<31>) is set, the interface will send MY_CYCLE and READ during P3, to the DRAM controller section of the FMDC in order to signal an octaword read request. During P4-P6 the interface will monitor the bus. If there was no refresh in progress at the time of the octaword read request, the interface will receive the READ_DONE signal from the DRAM controller in P6. During P7 the interface will assert its MBRQ, read status, and the first long word. During P8-P10 the interface will put the remaining long words on the MDAL. At the end of P9 the interface will remove its MBRQ.

If there had already been a refresh in progress at the time of the octaword request, then the READ_DONE signal will be delayed to a later cycle in order to allow completion of the refresh. In this case, WAIT status and MBRQ will be asserted during P7 and during each of the following cycles until the interface receives READ_DONE. In the four cycles following READ_DONE, read status, read data, and MBRQ will be asserted.

The maximum number of WAIT cycles that can be generated as a result of an in progress refresh during a memory read is **TBD**.

9.4.3.2. Memory Space Read (shared)

During P2 of an M-bus transaction, with MDAL bit 31 unasserted, the memory will interpret a READ or READI command as a memory space read. The memory M-bus interface will then latch the address and decode the upper bits to determine, based on the base address reg and the memory size, if the address falls within its domain. If the address is "owned", and the MEMSPEN bit (BASEADDR<31>) is set, the interface will send MY_CYCLE and READ during P3, to the DRAM controller section of the FMDC in order to signal an octaword read request. During P4-P6 the interface will monitor the bus. If there was no refresh in progress at the time of the octaword read request, the interface will receive the READ_DONE signal from the DRAM controller in P6.

In the case of a shared cycle, the interface will see a MBRQ during P6. The MBRQ indicates that one of the caches will supply read data for this transaction. The memory's M-bus interface will not drive the M-bus. It will allow the requesting cache to drive the read data, status, and MBRQ. Also, in the case of a shared read, The M-bus interface will monitor the M-Bus/CACHE transaction, as opposed to the DRAM controller response, in order to determine the end of transaction.

9.4.3.3. Memory Space Write (unshared)

During P2 of an M-bus transaction, with MDAL bit 31 unasserted, the memory will interpret a WRITE, WRITET or WRITEU command as a memory space write. The memory M-bus interface will then latch the address and decode the upper bits to determine, based on the base address reg and the memory size, if the address falls within its domain. If the address is "owned," and the MEMSPEN bit (BASEADDR<31>) is set, the interface will send MY_CYCLE and WRITE during P3, to the DRAM controller section of the FMDC in order to signal an octaword write request. In cycle P3-P6 the M-bus interface will transfer MDAL<31:0> into the data path section of the FMDC. Also in cycles P3-P6, the MDATINV signal will be monitored in order to determine if invalid write data is being presented. (see error strategy) In P6 only, the memory will assert its MBRQ to indicate a bus response.

If there was no refresh in progress at the time of the octaword write request, the interface will receive the WRITE_IN_PROG signal from the DRAM controller in cycle P3.

If there had already been a refresh in progress at the time of the octaword request, then the WRITE_IN_PROG signal will be delayed to a later cycle until completion of the refresh. In this case, depending on how many cycles WRITE_IN_PROG has been delayed, the MBUSY signal will be asserted in order to guarantee that latched data cannot be overwritten by another possible write transaction to the same memory.

9.4.3.4. Memory Space Write (shared)

The shared memory space write is handled identically to the way unshared memory space writes are handled.

9.4.3.5. Interlocked Transactions

Interlocked READ and WRITE transactions are treated, by the memory, exactly the same way as normal READ and WRITE transactions.

9.4.3.6. I/O Read Transaction

During P2 of an M-bus transaction, with MDAL bit 31 asserted, the memory will interpret a READ command as an I/O space read. During P3, the memory M-bus interface will decode the address bits to determine, based on the MID backplane slot code, if the address falls within the upper 16 longwords of its 32 mbyte I/O domain. If the address is "owned," the interface will prepare to drive the M-bus starting in P4 with its MBRQ, MSTATUS, and I/O read data on the MDAL lines.

9.4.3.7. I/O Write Transaction

During P2 of an M-bus transaction, with MDAL bit 31 asserted, the memory will interpret a WRITE command as an I/O space write. During P3, the memory M-bus interface will decode the address bits to determine, based on the MID backplane slot code, if the address falls within the upper 16 longwords of its 32 mbyte I/O domain. Also, during P3, the interface will strobe data from the MDAL and use the MCMD lines (mask) to determine which bytes are to be accessed. If the address is "owned," the interface will prepare to drive the M-bus starting in P4 with its MBRQ, and the MSTATUS with wait or write status.

9.4.3.8. Un-owned Transactions

During P3 of an M-bus transaction, if the Memory M-bus interface determines that the transaction is un-owned, (not within its address space) the interface will take the opportunity to attempt to do a Hidden Refresh. If, at that time the refresh window is open, then the DRAM controller will perform a refresh cycle.

The Memory M-bus interface will monitor all un-owned transactions for bus parity, correct protocol, and to keep in sync with the rest of the system.

9.4.3.9. No Slave Response

Lack of slave response during valid transactions will be detected by the memory interface during P4 of I/O space transactions, P4 of interrupt acknowledge transactions, and P7 of memory space read transactions. The lack of an MBRQ on the bus during any of the above cycles will cause the memory interface to recognize that the transaction has terminated and force it immediately to the idle state.

9.4.4. DRAM Controller

READ and WRITE cycles will consist of octaword data transfers which operate DRAMs using fast page mode timing protocol. Seventy-two bits of data (quadword + ECC) are written to and read from the DRAMs and transferred on the M-bus as four longwords, one cycle apart. Refresh is performed using CAS before RAS timing protocol.

READ and WRITE cycles assume that the addresses are asserted for the duration of P2. DRAM READ/WRITE timing state 0 will begin during P3.

Timing state 0 is an idle state in which the DRAM state machine determines what task to perform next: READ, WRITE, HIDDEN REFRESH, FORCED REFRESH, or remain idle.

During timing state 0, the DRAM control will remain in a WAIT state until a refresh request is received from the refresh state machine or read or write request is received from the M-bus state machine.

The DRAM state machine attaches three levels of priority to cycle requests. The highest priority cycle request is a FORCED REFRESH request. A DRAM access request, READ or WRITE, is second priority, and HIDDEN REFRESH is third priority.

The refresh state machine issues a FORCED REFRESH request when DRAMs have not been refreshed for 14 micro-seconds. (This value is subject to change.) If less than 14 micro-seconds have expired, this state machine will indicate that the "refresh window" is "open" or "closed." The open or closed refresh window is a flag which the refresh state machine sets to prevent refresh operations from occurring too frequently. If the cycle requested by the bus master does not require the memory module, the M-bus state machine will issue a HIDDEN REFRESH request to the DRAM state machine. The DRAM state machine will then perform a HIDDEN REFRESH if its refresh window is open and no higher priority cycle requests exist. However, if the refresh window is closed, the DRAM state machine will remain idle.

If a RAM access cycle is requested, RAS will be asserted at the conclusion of time 0 and either a READ or a WRITE cycle will begin during time 1.

During a READ cycle, COLUMN ADDRESSES will be selected via MUX ADDR and CAS will be asserted at time 1. QUADWORD 1 will be pre-latched at time 2. The column address will be incremented by one at time 3 and a DATA READY signal will be issued to the M-bus state machine. QUADWORD 2 will be pre-latched at time 4. CAS and RAS will need two cycles of precharge time after being negated during time 5.

During a WRITE cycle, RAS will be asserted and the first longword will be latched at time 0. The WRITE signal will be asserted at time 1, and the COLUMN ADDRESSES will be selected via MUX ADDR after a delay. LONGWORD 2 is also latched at time 1, and check bit generation is started for QUADWORD 1. LONGWORDs 3 and 4 are latched at times 2 and 3, respectively. CAS is asserted at time 3, and the DRAM state machine writes the first quadword at time 4. The column address is then incremented at time 5 and the second quadword is then strobed in. WRT_DONE will be asserted at time 5 for the M-bus protocol state machine, and all DRAM timing signals will be negated at time 7.

During refresh cycles, all CAS signals will be asserted at time 1, followed by all RAS signals at time 2. RAS and CAS will remain asserted until time 4.

9.4.5. Error Strategy

The following error strategy will be implemented with regard to M-bus errors or internal memory module errors. For failures during self-test, see Section 9.9.

9.4.5.1. M-Bus Errors

The M-bus class of errors are either bus parity errors or bus protocol errors. These errors are detectable by any of the M-bus interfaces. Any M-bus error detected by the memory will result in the assertion of the MABORT signal for eight cycles.

When MABORT is asserted for any reason, the memory will abort any DRAM cycle that has not yet begun, or complete a DRAM cycle to the point that DRAM timing may be terminated without the loss of data.

The Memory M-bus interface will check parity on the MCMD, MSTATUS, and MDAL at appropriate times during a transaction. When the FMDC detects a bus parity error, it asserts MABORT. The following table, taken from the M-bus specification, illustrates the proper times to check each type of parity for no-wait transactions. The letters "C", "S", or "D" are meant to represent the three parity types.

Table 9-8: M-Bus Parity Checking

Transaction	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
Memory Read		CD					SD	SD	SD	SD
Memory Write		CD	CD	CD	CD	CD				
I/O Read		CD	C	SD						
I/O Write		CD	CD	S						
Interrupt Ack		CD			SD					

For wait transactions, MSTATUS parity and MDAL parity as shown in the table will extend out into later cycles.MDAL parity errors will be ignored during all cycles in which the MSTATUS lines indicate "WAIT".

Multiple MBRQ signals on the bus during non-arbitration cycles will result in MABORT. The following table illustrates which cycles are covered for multiple MBRQ errors. The table uses an "M" to signify cycles with master asserted requests and "S" to signify cycles with slave asserted requests. The cycles shown are for non-wait transactions. If wait cycles occur, then slaves will assert their MBRQ for additional numbers of cycles equal to the number of waits.

Table 9-9: Cycles Covered for Multiple MBRQ Errors

Transaction	P1	P2	P3	P4	P5	P6	P7	P8	P9
Memory Read		M	M	M	M		S	S	S
Memory Write		M	M	M	M	S*			
I/O Read		M	M	S					
I/O Write		M	M	S					
Interrupt Ack		M	M		S				

* (memory victim writes only)

The memory will assert MABORT if it detects a code other than the seven valid command codes on the MCMD lines During the command cycle of a transaction, cycle P2.

When the memory detects more than 256 cycles of WAIT or MBUSY status on the bus, it will assert MABORT.

9.4.5.2. Internal Memory Errors

The internal memory class of errors are, by definition, the result of some internal memory failure. Coverage of these errors is intended mainly to prevent data corruption but also to provide valuable information for debug, failure analysis, and reliability.

Single Bit Errors (SBE) are generally the result of a DRAM "soft" error. Single bit errors are covered for both write and read transactions by the ECC circuitry. All SBE's (hard or soft) on either data or check bits will be corrected when read. The SBE will be reported to the M-bus during a read transaction as "corrected" on the MSTATUS lines for two consecutive long words. (ECC is done on 64 bits) The Syndrome bits will be available in an I/O register, if needed, to determine which bit was corrected.

Multiple Bit Errors (MBE) can be the result of two or more data bits in error, or an address parity error. MBE's are detected but not corrected. They are reported to the M-bus by asserting the MDATINV signal during a read transaction for two consecutive longwords. Syndromes cannot identify which bits were in error for a MBE but special syndrome codes can identify the occurrence of an address parity error.

The memory will not store the fact that it detected SBEs or MBEs in the case of shared read transactions. The assumption is that the error will be detected on subsequent reads to the same address.

A no-response from the DRAM state machine will cause the M-bus to time-out. If the memory interface does not receive DATA_RDY, in the case of a read or WRITE_IN_PROG in the cae of a write, from the

DRAM state machine, the M-bus interface will assert "WAIT" on MSTATUS or the signal MBUSY to force a SLAVE_TIMEOUT.

9.5. Module Power Requirements

The memory module will require one power source, +5V. The maximum memory power will be required during a read cycle.

The following assumptions have been made to calculate memory module power requirements: read cycle time is 650 ns, CAS pulse width is 80 ns, refresh cycle time is 260 ns, and refresh period is 14 microseconds.

Table 9-10: Memory Module Power Requirements

Module Size	Operating Mode	Current (Amps)/ Power (Watts) Minimum	Typical	Maximum
8 Mbyte	Standby	.84/4.0	1.4/7.2	2.1/11.2
8 Mbyte	Active	2.0/9.7	3.2/16.2	4.0/21.1
16 Mbyte	Standby	1.1/5.4	1.9/9.5	2.8/14.5
16 Mbyte	Active	2.3/11.0	3.7/18.4	4.6/24.3
32 Mbyte (w/1 Mbit DRAMs)	Standby	1.6/7.5	2.6/13.0	3.7/19.2
32 Mbyte (w/1 Mbit DRAMs)	Active	2.8/13.3	4.4/21.9	5.5/29.1

9.6. Module MTBF

This section presents the failure rate characteristics assumed for the 1M x 1 CMOS DRAMs and summary of the MTBF results obtained using these assumptions. The MTBF numbers were calculated using a Monte Carlo model of the DRAM failure characteristics. The DRAM failure characteristics are shown below:

ECC failure rates are based on the following DRAM failure distribution:

Table 9-11: DRAM Failure Distribution

Failure Type	% of Failures
Single bit	45
Row	25
Column	15
Whole	10
Interactive	5

The table below describes failure rates for the 16Mbyte module. These failure rates are based on the following mos DRAM failure rates:

Ground fixed	=	0.5 failures/million hours
Ground benign	=	0.3 failures/million hours
Soft errors	=	3.0 failures/million hours

Table 9-12: Module Component Failure Rates (Failures/Million Hours)

Type	GB	GF
IC'S and Discrete	.9	2.7
MOS RAMS (Hard)	43.2	72.0
MOS RAMS (Soft)	432.0	720.0
Total Module (No ECC)	44.1	74.7
Total Module (ECC)	10.5	18.5
MTBF Summary (Hours)		
Total Module (No ECC)	26,676	13,387
Total Module (ECC)	95,374	53,916

9.7. Register Description

The following sections describe the I/O registers implemented on the Firefox memory module FMDC gate array.

9.7.1. FMDC Registers

The FMDC supports a total of 16 internal registers. These registers implement the following:

- M-bus module type identification
- M-bus error detection and status
- M-bus error control signal log
- M-bus error address signal log
- M-bus error data signal log
- Control and status of the FMDC
- Memory space base address offset
- Memory ECC error address retention (quadword 0)
- Memory ECC error address retention (quadword 1)
- Memory ECC status (quadword 0)
- Memory ECC status (quadword 1)
- Memory section in error (self test results)
- M-bus state machine signature register
- DRAM state machine signature register
- Self test state machine signature register
- Diagnostic/self test LEDs
- M-bus state machine signature expect register
- DRAM state machine signature expect register
- Self test state machine signature expect register

9.7.2. FMDC Register Map

Table 9-13 lists the FMDC registers and their function. The address field of the table shows the low order 24 bits of M-bus address necessary to access a register. The XX address bits are related to the M-bus slot position in which the memory module resides and are described in Table 9-14.

Table 9-13: FMDC Register Map

FMDC Register Offsets			
Name	**Address**	**R/W**	**Description**
MODTYPE	XXFFFFFC#16	R	Module type register
BUSCSR	XXFFFFF8#16	R/W	M-bus error status register
BUSCTL	XXFFFFF4#16	R/W	M-bus error control signal log register
BUSADR	XXFFFFF0#16	R/W	M-bus error address signal log register
BUSDAT	XXFFFFEC#16	R/W	M-bus error data signal log register
FMDCSR	XXFFFFE8#16	R/W	FMDC control/status register
BASEADDR	XXFFFFE4#16	R/W	Memory space base address register
ECCADDR0	XXFFFFE0#16	R	Memory ECC error address reg.(QW0)
ECCADDR1	XXFFFFDC#16	R	Memory ECC error address reg.(QW1)
ECCSYND0	XXFFFFD8#16	R/W	Memory ECC error status reg.(QW0)
ECCSYND1	XXFFFFD4#16	R/W	Memory ECC error status reg.(QW1)
MSECTERR	XXFFFFD0#16	R	Memory section had error register
MBUSSIG	XXFFFFCC#16	R/W	M-bus control signature register
DRAMSIG	XXFFFFC8#16	R/W	DRAM control signature register
SELFSIG	XXFFFFC4#16	R/W	Self test signature register
LEDLATCH	XXFFFFC0#16	R/W	Diagnostic/self test LED latch
EXP_MBUSSIG	XXFFFFBC#16	R	M-bus control signature expect register
EXP_DRAMSIG	XXFFFFB8#16	R	DRAM control signature expect register
EXP_SELFSIG	XXFFFFB4#16	R	Self test signature expect register

Table 9-14: FMDC I/O Register Base Addresses

Slot	M-bus Address
0	91000000#16
1	93000000#16
2	95000000#16
3	97000000#16
4	99000000#16
5	9B000000#16
6	9D000000#16
7	9F000000#16

9.7.3. FMDC Register Summary

9.7.3.1. MODTYPE Register

The MODTYPE register is an 32-bit read-only register, accessible through I/O space to any processor in the Firefox system. It indicates the class of the module, class specific information, the interface chip type, and the interface chip revision. Figure 9-3 shows the bit definitions for the MODTYPE register which are defined below:

```
             Name: MODTYPE       Address: XXFFFFFC#16        Access: R

          3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
          1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
         +---------------------------------------------------------------+
         |   REVISION    |0 0 0 0 0 0 1 0|0 0 0 0 0|0|0 1|0 0 0 1 0 0 0 0|
         +-------^---------------^-----------------^--^---------^--------+
     MODTYPE     |               |                 |  |         |
                 |               |                 |  |         |
                 |               |                 |  |         |
     Interface Chip Revision     |                 |  |         |
                                 |                 |  |         |
     Interface Chip Type --------+                 |  |         |
                                                   |  |         |
     RAM Type -------------------------------------+  |         |
                                                      |         |
     Number of Banks ---------------------------------+         |
                                                                |
     Class -----------------------------------------------------+
```

Figure 9-3: MODTYPE Register

MODTYPE<7:0>CLASS (R)

A value of 10#16 in this bit field indicates that a memory module is present.

MODTYPE<9:8>NUMBER_OF_BANKS (R)

Bits 8 and 9 are an encoded bit field used to indicate the number or DRAM banks present on this module. The number of DRAM banks present can be one, two, or four and are represented by encoding <9:8> as 00#2, 01#2 and 11#2 respectively.

MODTYPE<10>4M_RAM_TYPE (R)

A zëro in this bit indicates that the module contains 1 M × 1 DRAMs, and a one in bit 10 indicates that the module contains 4 M × 1 DRAMs.

Bits <10:8> can be used to determine the array size as outlined in the DRAM array section of the memory spec. For example, a 001#2 in MODTYPE<10:8> would indicate that 1 M × 1 chips and two banks of DRAMs were present; hence, the module size is 16 MB. See Table 9-15 for examples of actual instances of this encoding.

Table 9-15: MODTYPE Memory Module Size Encoding

MODTYPE	<10>	<9>	<8>	# of Banks	DRAM Type	Array Size
	0	0	0	one	1 M × 1	8 MB
	0	0	1	two	1 M × 1	16 MB
	0	1	0	Reserved	1 M × 1	Reserved
	0	1	1	four	1 M × 1	32 MB
	1	0	0	one	4 M × 1	32 MB
	1	0	1	two	4 M × 1	64 MB
	1	1	0	Reserved	4 M × 1	Reserved
	1	1	1	four	4 M × 1	128 MB

MODTYPE<23:16>INT_CHIP_TYPE (R)

A value of 02#16 in this bit field indicates that an FMDC interface chip is present on this module.

MODTYPE<31:24>INT_CHIP_REV (R)

The value in this bit field indicates the revision level of the FMDC interface chip on this module.

9.7.3.2. BUSCSR Register

The BUSCSR register is a read/write register that maintains M-bus error status, and controls M-bus error logging. Figure 9-4 shows the bit definitions for the BUSCSR register. All bits of the BUSCSR are active low. The FMDC logs errors in the BUSCSR whenever the FROZEN bit is set by clearing the corresponding status bit. Whenever the FMDC clears a status bit, it also clears the FROZEN bit. When the FROZEN bit is clear, the FMDC does not alter the value of the BUSCSR. That is, the BUSCSR saves the first error event. If simultaneous errors occur, and error logging is enabled, the FMDC clears multiple status bits. To enable error logging, write FFFFFFFF#16 to BUSCSR. The result of simultaneous I/O space writes to BUSCSR and error events is unpredictable.

```
               Name: BUSCSR       Address: XXFFFFF8#16        Access: RW
                     3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
                     1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
                    +-+-+-+-+-+-+-+-+-+-+-------------------------------------------+
                    |F|S|I|R|R|M|M|M|R|W|                 RESERVED                  |
                    +^+^+^+^+^+^+^+^+^+^+---------------------^---------------------+
    BUSCSR           | | | | | | | | | |                      |
                     | | | | | | | | | |                      |
                     | | | | | | | | | |                      |
    FROZEN ----------+ | | | | | | | | |                      |
    SLAVE_ARB_ERR -----+ | | | | | | | |                      |
    INVALID_MCMD --------+ | | | | | | |                      |
    RESERVED --------------+ | | | | | |                      |
    RESERVED ----------------+ | | | | |                      |
    MDAL_PARITY_ERR -----------+ | | | |                      |
    MSTAT_PARITY_ERR ------------+ | | |                      |
    MCMD_PARITY_ERR ---------------+ | |                      |
    RESERVED ------------------------+ |                      |
    SLAVE_TIMEOUT ---------------------+                      |
    RESERVED -------------------------------------------------+
```

Figure 9-4: BUSCSR Register

BUSCSR<21:0>RESERVED

These bits are reserved for future use. They must always be written with zero's for compatibility with future extensions.

BUSCSR<22>SLAVE_TIMEOUT (RW) M-bus Slave Timeout

SLAVE_TIMEOUT is cleared when a bus slave specifies WAIT status or MBUSY for more than 256 M-bus cycles. The FROZEN bit is cleared together with the SLAVE_TIMEOUT bit. The FMDC also generates a MABORT sequence when it clears SLAVE_TIMEOUT. System reset clears SLAVE_TIMEOUT.

BUSCSR<23>RESERVED (R)

This bit is reserved for future use. It must always be written with a 0 for compatibility with future extensions. It will always be read as a 0.

BUSCSR<24>MCMD_PARITY_ERR (RW) M-bus MCMD Parity Error

MCMD_PARITY_ERR is cleared when a parity error occurs on the MCMD signals. The FMDC checks MCMD parity the cycle after the value is on the M-bus, that is: P3; memory write P4, P5, P6, first P7; I/O read first P4; and I/O write first P4. The FMDC also generates a MABORT sequence when it clears MCMD_PARITY_ERR. The FROZEN bit is cleared together with the MCMD_PARITY_ERR bit. System reset clears MCMD_PARITY_ERR.

BUSCSR<25>MSTAT_PARITY_ERR (RW) M-bus MSTATUS Parity Error

MSTAT_PARITY_ERR is cleared when a parity error occurs on the MSTATUS signals. The FMDC checks MSTATUS parity the cycle after the value is on the M-bus, that is: memory read P8, P9, P10, P11; I/O read P5; I/O write P5; and interrupt acknowledge P6. The FMDC also generates a MABORT sequence when it clears MSTAT_PARITY_ERR. The FROZEN bit is cleared together with the MSTAT_PARITY_ERR bit. System reset clears MSTAT_PARITY_ERR.

BUSCSR<26>MDAL_PARITY_ERR (RW) M-Bus MDAL Parity Error

MDAL_PARITY_ERR is cleared when a parity error occurs on the MDAL. The FMDC checks MDAL parity the cycle after the value is on the M-bus, that is: P3; memory read P8, P9, P10, P11; memory write P4, P5, P6, first P7; I/O read P5; I/O write first P4; and interrupt acknowledge P6. The FMDC also generates a MABORT sequence when it clears MDAL_PARITY_ERR. The FROZEN bit is cleared together with the MDAL_PARITY_ERR bit. System reset clears MDAL_PARITY_ERR.

BUSCSR<28:27>RESERVED

These bits are reserved for future use. They must always be written with zero's for compatibility with future extensions.

BUSCSR<29>INVALID_MCMD (RW) M-Bus Invalid MCMD Error

INVALID_MCMD is cleared when the memory detects an invalid MCMD encoding during P2. The FMDC also generates a MABORT sequence when it clears iNvALID_MCMD. The FROZEN bit is cleared together with the INVALID_MCMD bit. System reset clears INVALID_MCMD.

BUSCSR<30>SLAVE_ARB_ERR (RW) M-Bus Slave Arbitration Error

SLAVE_ARB_ERR is cleared when the memory detects an M-bus arbitration error. Arbitration errors are either premature deassertion of an MBRQ signal when the FMDC is monitoring a transaction, or assertion of another MBRQ signal when the FMDC has its MBRQ signal asserted as the slave of a transaction. The FMDC also generates an MABORT sequence when it clears SLAVE_ARB_ERR. The FROZEN bit is cleared together with the SLAVE_ARB_ERR bit. System reset clears SLAVE_ARB_ERR.

BUSCSR<31>FROZEN (RW) BUSCSR Latch Is Frozen

When the memory detects an M-bus error, or an MABORT on the bus, the FROZEN bit is cleared and the BUSCSR, BUSCTL, BUSADR and BUSDAT registers are frozen. FROZEN bit may be used as an error summary bit for M-Bus errors. System reset clears FROZEN.

9.7.3.3. BUSCTL Register

The BUSCTL register is a read/write register that logs the value of M-bus control signals at the time the FMDC detects an error. Write access to the BUSCTL register is only for diagnostic testing. The result of I/O space writes to the BUSCTL register when error logging is enabled is unpredictable. Figure 9-5 shows the bit definitions for the BUSCTL register.

```
                Name: BUSCTL       Address: XXFFFFF4#16        Access: RW

             3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
             1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
            +-------+-+-+-----+-+-+-+-+-+-+-+---+-+-------+-+-+-------------+
            |  SMC  |R|S| PHZ |R|A|D|S|B|D|S|MS |C| MCMD  |Q|R|    MBRM     |
            +---^---+^+^+--^--+^+^+^+^+^+^+^+-^-+^+---^---+^+^+------^------+
BUSCTL          |    | |   |   | | | | | | |  |  |    |    | |              |
                |    | |   |   | | | | | | |  |  |    |    | |              |
SVDMCMD --------+    | |   |   | | | | | | |  |  |    |    | |              |
RESERVED ------------+ |   |   | | | | | | |  |  |    |    | |              |
SLAVE -----------------+   |   | | | | | | |  |  |    |    | |              |
PHASE ---------------------+   | | | | | | |  |  |    |    | |              |
RESERVED ----------------------+ | | | | | |  |  |    |    | |              |
MABORT --------------------------+ | | | | |  |  |    |    | |              |
MDATINV ---------------------------+ | | | |  |  |    |    | |              |
MSHARED -----------------------------+ | | |  |  |    |    | |              |
MBUSY ---------------------------------+ | |  |  |    |    | |              |
MDPAR -----------------------------------+ |  |  |    |    | |              |
MSPAR -------------------------------------+  |  |    |    | |              |
MSTATUS --------------------------------------+  |    |    | |              |
MCPAR -------------------------------------------+    |    | |              |
MCMD -------------------------------------------------+    | |              |
MBRQ ------------------------------------------------------+ |              |
RESERVED ----------------------------------------------------+              |
MBRM -----------------------------------------------------------------------+
```

Figure 9-5: BUSCTL Register

BUSCTL<6:0>MBRM (RW) M-bus MBRM Signals

The FMDC continuously updates MBRM from the M-bus MBRM signals whenever BUSCSR<FROZEN> is set.

BUSCTL<7>RESERVED

This bit is reserved for future use. It must always be written with zero's for compatibility with future extensions.

BUSCTL<8>MBRQ (RW) M-bus MBRQ Signal

The FMDC continuously updates MBRQ from the FMDC MYMBRQ output pin whenever BUSCSR<FROZEN> is set.

BUSCTL<12:9>MCMD (RW) M-bus MCMD Signals

The FMDC updates MCMD from the M-bus MCMD signals whenever it checks parity on the MCMD signals, and BUSCSR<FROZEN> is set.

BUSCTL<13>MCPAR (RW) M-bus MCPAR Signal

The FMDC updates MCPAR from the M-bus MCPAR signal whenever it checks parity on the MCMD signals, and BUSCSR<FROZEN> is set.

BUSCTL<15:14>MSTATUS (RW) M-bus MSTATUS Signals

The FMDC updates MSTATUS from the M-bus MSTATUS signals whenever it checks parity on the MSTATUS signals, and BUSCSR<FROZEN> is set.

BUSCTL<16>MSPAR (RW) M-bus MSPAR Signal

The FMDC updates MSPAR from the M-bus MSPAR signal whenever it checks parity on the MSTATUS signals, and BUSCSR<FROZEN> is set.

BUSCTL<17>MDPAR (RW) M-bus MDPAR Signal

The FMDC updates MDPAR from the M-bus MDPAR signal whenever it checks parity on the MDAL signals, and BUSCSR<FROZEN> is set.

BUSCTL<18)MBUSY (RW) M-bus MBUSY Signal

The FMDC continuously updates MBUSY from the M-bus MBUSY signal whenever BUSCSR<FROZEN> is set.

BUSCTL<19)MSHARED (RW) M-bus MSHARED Signal

The FMDC continuously updates MSHARED from the M-bus MSHARED signal whenever BUSCSR<FROZEN> is set.

BUSCTL<20)MDATINV (RW) M-bus MDATINV Signal

The FMDC continuously updates MDATINV from the M-bus MDATINV signal whenever BUSCSR<FROZEN> is set.

BUSCTL<21)MABORT (RW) M-bus MABORT Signal

The FMDC updates MABORT from the M-bus MABORT signal whenever BUSCSR<FROZEN> is set.

BUSCTL<22>RESERVED

This bit is reserved for future use. It must always be written with zero's for compatibility with future extensions.

BUSCTL<25:23>PHASE (RW) M-bus Transaction Phase

The FMDC continuously updates PHASE from the M-bus state machine transaction phase whenever BUSCSR<FROZEN> is set. Table 9-16 shows the encoding of PHASE as a function of the M-bus transaction phase.

Table 9-16: BUSCTL M-bus Transaction Phase Encoding

M-bus	Phase
P1	0
P2	1
P3	2
P4	3
P5	4
P6	5
P7	6
P8	7
P9	7
P10	7

BUSCTL<26)SLAVE (RW) M-bus SLAVE

The FMDC continuously updates SLAVE from the M-bus state machine slave mode whenever BUSCSR<FROZEN> is set.

BUSCTL<27>RESERVED

This bit is reserved for future use. It must always be written with zero's for compatibility with future extensions.

BUSCTL<31:28>SVDMCMD(RW) M-bus Saved MCMD Signals

The FMDC updates SVDMCMD from the P2 value of the M-bus MCMD signals during P3 of every transaction whenever <FROZEN> is set.

9.7.3.4. BUSADR Register

BUSADR is a read/write register that latches the value of MDAL on the M-bus during P2 of every transaction, unless BUSCSR<31> is clear. The BUSADR register should only be written during diagnostic self test or error logging information is unpredictable. If the BUSCSR indicates a MDAL parity error, parity should be calculated on BUSADR and compared with the MDAL parity bit saved in the BUSCTL register to determine whether or not BUSADR is valid. Figure 9-6 shows the bit definitions for the BUSADR register.

```
                Name: BUSADR        Address: XXFFFFF0#16         Access: RW


          3
          1                                                              0
         +----------------------------------------------------------------+
         |                            ADDRESS                             |
         +-----------------------------^----------------------------------+
    BUSADR                             |
                                       |
    ADDRESS ---------------------------+
```

Figure 9-6: BUSADR Register

BUSADR<31:0>ADDRESS (R) M-bus Error Address

The BUSADR register records the value on the MDAL signals during cycle P2 of every M-bus transaction. When the BUSCSR<FROZEN> bit is cleared, contents of the BUSADR register is frozen. When the BUSCSR<FROZEN> bit is set, the BUSADR register may be used as a read/write register to verify its operation. When the BUSCSR<FROZEN> bit is clear, writing the BUSADR register produces unpredictable results.

9.7.3.5. BUSDAT Register

The BUSDAT register is a read/write register that maintains the last M-bus MDAL value. Write access to the BUSDAT register is only for diagnostic testing; the result of I/O space writes to the BUSDAT Register when error logging is enabled is unpredictable. Figure 9-7 shows the bit definitions for the BUSDAT register.

```
          Name: BUSDAT       Address: XXFFFFEC#16         Access: RW

      3
      1                                                            0
     +-----------------------------------------------------------------+
     |                          DATA                                   |
     +----------------------------^------------------------------------+
BUSDAT                            |
                                  |
DATA -----------------------------+
```

Figure 9-7: BUSDAT Register

BUSDAT<31:0>DATA (RW) M-bus Error Data

> The FMDC updates DATA from the MDAL signals whenever the FMDC checks parity on the MDAL signals, and BUSCSR<FROZEN> is set.

9.7.3.6. FMDCSR Register

The FMDCSR Register is an 32-bit read/write register that holds control and status information for the memory module. The bits and sub-fields of this register are defined in figure 4-6 below :

```
          Name: FMDCSR       Address: XXFFFFE8#16        Access: RW

         3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
         1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
        +-+-+-----------+-+-+-----+---+---+-+-+-+-+-+-+-+---------------+
        |E|O| Reserved  |I|I| FEC |FES|ECC|R|R|R|R|D|S|S|   RAS CNTR    |
        +^+^+-----------+^+^+--^--+-^-+-^-+^+^+^+^+^+^+^+-------^-------+
FMDCSR   | |             | |   |    |   |  | | | | | | |        |
         | |             | |   |    |   |  | | | | | | |        |
         | |             | |   |    |   |  | | | | | | |        |
         | |             | |   |    |   |  | | | | | | |        |
         | |             | |   |    |   |  | | | | | | |        |
Error    | |             | |   |    |   |  | | | | | | |        |
Flag ----+ |             | |   |    |   |  | | | | | | |        |
Summary    |             | |   |    |   |  | | | | | | |        |
           |             | |   |    |   |  | | | | | | |        |
On Line ---+             | |   |    |   |  | | | | | | |        |
                         | |   |    |   |  | | | | | | |        |
INH_SBE_MSECTERR_LOG ----+ |   |    |   |  | | | | | | |        |
                           |   |    |   |  | | | | | | |        |
INH_SBE_REPORT ------------+   |    |   |  | | | | | | |        |
                               |    |   |  | | | | | | |        |
Force Error Category ----------+    |   |  | | | | | | |        |
                                    |   |  | | | | | | |        |
Force Error Sub-Category -----------+   |  | | | | | | |        |
                                        |  | | | | | | |        |
ECC Diagnostic Mode --------------------+  | | | | | | |        |
                                           | | | | | | |        |
Diagnostic Refresh Start ------------------+ | | | | | |        |
                                             | | | | | |        |
Refresh Period Select -----------------------+ | | | | |        |
                                               | | | | |        |
Disable Refresh -------------------------------+ | | | |        |
                                                 | | | |        |
RESERVED ----------------------------------------+ | | |        |
                                                   | | |        |
Data to Check Bits --------------------------------+ | |        |
                                                     | |        |
Self Test Complete ----------------------------------+ |        |
                                                       |        |
Self Test Start ---------------------------------------+        |
                                                                |
RAS/Refresh Counter --------------------------------------------+
```

Figure 9-8: FMDCSR Register

FMDCSR<7:0>RAS_REFRESH_COUNTER (RW)

This eight bit, continuously running counter will count refresh cycles as a check of the REFRESH control logic and the refresh rate. By loading the counter with 0's, the console or operating software can expect to read a value within a specified range after a given time interval. Although the memory module will count every refresh cycle, the value stored in the counter will be scaled down to 8 bits. The current plan is to increment the counter on every eighth refresh cycle. The value range will be defined at a later time. System reset will not clear the counter and will have no effect on the counter value.

FMDCSR<8>SELF_TEST_START (RW)

Bit 8 tells the array to begin DRAM self test. The bit is set by writing a one, and writing a zero has no effect. The self test start bit is cleared by the array on system reset and at the completion of self test. Once self test is underway, the memory module is not available and will not respond to memory requests until self test is complete. The MODULE_ONLINE bit (FMDCSR<30>) will be negated for the duration of self test even if MEMSPEN (BASEADDR<31>) is asserted.

FMDCSR<9>SELF_TEST_COMPLETE (R)

Bit 9 is used by the array to indicate that DRAM self test has been run and is complete.The self test complete bit is cleared by the array on system reset and at the beginning of DRAM self test.

FMDCSR<10>DATA_TO_CHECK_BITS (RW)

While enabled, the DTCB signal allows data to be written to and read directly from the ECC check bit DRAMs. During an octaword WRITE cycle with this bit asserted, the low order byte of longwords 0 and 2 will be written as the check bits of quadword 0 and 1 respectively; (MDAL<7> --> CB<7>, MDAL<6> --> CB<6>,..., MDAL<0> --> CB<0>). This will be done in lieu of writing the check bits that the ECC circuit normally calculates based on the data. Also, the same bytes are written into their normal locations in the quadwords.

During octaword READs with this bit set, normal ECC checking and error flagging is disabled. Also, the read data from the ECC check bit DRAMs for quadwords 0 and 1 are routed to the low order bytes of longwords 0 and 2 respectively (CB<7> --> MDAL<7>, CB<6> --> MDAL<6>,..., CB<0> --> MDAL<0>).

This bit is cleared by system reset.

FMDCSR<11>Reserved (R)

Bit 11 will be read as zero, and writing a one to this bit will have no effect.

FMDCSR<12>DISABLE_REFRESH (RW)

When set, the disable refresh bit disables the automatic refresh features of the memory module. There is no guarantee that data stored in the array will be preserved while bit 12 is asserted unless all rows of the DRAMs are refreshed (ie: read or rewritten) every 8 milliseconds. To avoid this, each DRAM row should be refreshed, on average, every 15 microseconds because there are 512 rows in the each 1 M x 1 chip.

FMDCSR<13>RPS (RW) REFRESH PERIOD SELECT

The RPS bit allows the refresh state machine to optimize its refresh counter to different clock periods planned for the M-bus. On power up, the memory will clear the bit and operate assuming that a 120 ns clock is being used. If a clock period of less than 120 ns is used, the refresh requirements of the dynamic RAMS will still be met; however, there is a slightly higher chance of a refresh slowing down a memory transaction. The CPU may select a slower refresh rate by writing a one to this bit. When the RPS bit is set to one, the memory will assume that a 72 ns clock is being used.

It is important to note that selecting a different refresh rate will alter the expected value found in the RAS/REFRESH counter, FMDCSR<7:0>, when checking the refresh rate.

FMDCSR<14>DIAGNOSTIC_REFRESH_START (W)

The DRS bit allows memory manufacturing to externally issue a refresh command. This bit is used in a test environment in conjunction with assertion of the FMDCSR<12>DISABLE_REFRESH bit) to force a refresh cycle to occur. Its intention is to allow the number of clock cycles that a memory operation takes to be deterministic by removing REFRESH as a nondeterministic event. This command will have no effect unless the DISABLE_REFRESH bit (FMDCSR<12> is set.

FMDCSR<16:15>ECC_DIAG_MODE (RW) ECC Diagnostic Mode Bits

This bit field is used to select one of the three test modes available to functionally check the ECC circuitry. Table 9-17 shows the encoding for this field and Section 9.9 explains the purpose of each mode. Writing to this field causes it to be written with the values present on the MDAL <16:15> data lines. Reading this field causes its contents to be placed on the MDAL<16:15> data lines. System reset clears these bits.

Table 9-17: ECC_DIAG_MODE Bit Definitions

FMDCSR<16:15>	ECC Diagnostic Mode
00#2	Normal Operation
01#2	Diagnostic Mode 1 - Verify CB generation logic
10#2	Diagnostic Mode 2 - Verify detection and correction logic
11#2	Diagnostic Mode 3 - Verify MOS DRAM CB field

FMDCSR<18:17>FORCE_ERROR_SUB-CATEGORY (RW)

Bits 18 and 17 further decode the eight major force error category groups (defined by FMDCSR<21:19>) into one to four error types as indicated below in table 4-6. These bits are cleared by RESET and set to all one's by MABORT.

FMDCSR<21:19>FORCE_ERROR_CATEGORY (RW)

Bits 21 through 19 decode eight major force error categories as indicated below in Table 9-18. These bits are cleared by RESET and set to all one's by MABORT.

FMDCSR<21:17>FORCED_ERRORS

NORMAL OPERATION (00000 #2)

In this mode no errors will be forced.

ISOLATE (11111 #2)

In this mode, the memory will not be capable of generating additional MABORTS. An MABORT, generated internally or externally, will assert the isolate mode in order to ensure only a single MABORT (8 cycle) transaction.

Table 9-18: Force Error Definitions

FMDCSR <21:19>	Force Error Category	FMDCSR <18:17>	Force Error Sub-Category
000 #2	No Force Errors	XX #2	
001 #2	Address Parity	00 #2	BUS Address
		01 #2	Row Address
		10 #2	Column Address
		11 #2	Row & Column
010 #2	Data Parity	00 #2	Longword 0
		01 #2	Longword 1
		10 #2	Longword 2
		11 #2	Longword 3
011 #2	MDATINV	00 #2	Longword 0
		01 #2	Longword 1
		10 #2	Longword 2
		11 #2	Longword 3
100 #2	Time Out	00 #2	Reserved
		01 #2	WAIT
		10 #2	MBUSY
		11 #2	Reserved
101 #2	Slave Arb	00 #2	Reserved
		01 #2	Multiple MBRQ
		10 #2	Premature deassertion
		11 #2	Reserved
110 #2	MCMD	00 #2	Reserved
		01 #2	Parity Error
		10 #2	Reserved
		11 #2	Invalid MCMD
111 #2	MStatus	00 #2	Reserved
		01 #2	Parity Error
		10 #2	Reserved
	ISOLATE	11 #2	ISOLATE

FMDCSR<21:17>FORCED_ERRORS (cont.)

ADDRESS PARITY ERRORS (001XX #2) - Four types of address parity errors can be forced:

BUS ADDRESS (00100 #2) - Selection of this test will cause the MDPARITY bit to be inverted during P2 of memory space writes. During a memory space write to this memory, with this bit set, the memory will respond with MABORT, and no write to the DRAM will take place.

ROW ADDRESS (00101 #2) - This test mode will cause the memory to invert the row address parity bit feeding into the ECC checker during a READ cycle. The FMDC will assert MDATINV along with the read data. The memory will report the error information in the ECC address and syndrome registers upon completion of the cycle and indicate row address parity error on the syndromes.

COLUMN ADDRESS (00110 #2) - This test mode will cause the memory to invert the column address parity bit feeding into the ECC checker during a READ cycle. The FMDC will assert MDATINV along with the read data. The memory will report the error information in the ECC address and syndrome registers upon completion of the cycle and indicate column address parity error on the syndromes.

ROW AND COLUMN (00111 #2) - This test mode will cause the memory to invert the row and column address parity bit feeding into the ECC checker during a READ cycle. The FMDC will assert MDATINV along with the read data. The memory will report the error information in the ECC address and syndrome registers upon completion of the cycle and indicate both row and column address parity error on the syndromes.

DATA PARITY ERRORS (010XX) - This bit is intended for use in a write/ read test cycle and will cause the memory to invert MDPARITY bits (corresponding to each longword) during a WRITE cycle. Upon detection of a data parity error, the memory will invert the quadword's ECC bits. The DRAM state machine will complete the write cycle and the M-bus interface will assert MABORT. During the read cycle, memory will flag the invalid data by asserting MDATINV along with the invalid data words. See Section 9.4.5 for more information about parity checking.

MDATINV ERRORS (011XX) - When this diagnostic mode is set, subsequent memory space write transactions will force a MDATINV error to occur for the appropriate longword of an octaword write transaction. There are four separate modes (one for each longword) for this test. A memory space write transaction in one of these modes will emulate a MDATINV error during the appropriate longword. The effect of running this test is that the quadword will be "marked" with bad ECC indicating an uncorrectable error.

MDATINV LW0 (01100 #2) - Quadword 0 gets marked

MDATINV LW1 (01101 #2) - Quadword 0 gets marked

MDATINV LW2 (01110 #2) - Quadword 1 gets marked

MDATINV LW3 (01111 #2) - Quadword 1 gets marked

TIMEOUT ERRORS (100XX) - Two kinds of timeout errors are possible and will be detected by the memory:

WAIT (10001) - A WAIT timeout error will be emulated by preventing the DRAM state machine from responding to a read request from the M-bus state machine. During this test mode, the terminal count of the wait timeout counter will be changed to 128 #10 so that the memory will be guaranteed to be the first to respond with MABORT. The memory will flag the error by asserting BUSCSR<22> (SLAVE_TIMEOUT).

MBUSY (10010) - AN MBUSY timeout error will be emulated by preventing the DRAM state machine from responding to a write request from the M-bus state machine. During this test mode, the terminal count of the MBUSY timeout counter will be changed to 128 #10 so that the memory will be guaranteed to be the first to respond with MABORT. The memory will flag the error by asserting BUSCSR<22> (SLAVE_TIMEOUT).

SLAVE ARB (101XX) These modes will check the memory's ability to detect multiple MBRQ errors and master premature deassertion of MBRQ errors.

MULTIPLE MBRQ (10101) - This test will force a multiple MBRQ, and the memory will issue MABORT. During MRP7, while in this diagnostic mode, the memory will force a false ANYARB signal. This will cause an internal ARBERR to initiate MABORT.

NO MBRQ ON BUS (10110) - During MWP5, while in this diagnostic mode, the memory will inhibit the ANYARB signal in order to give a false indication that the master's MBRQ was lost. This will cause an internal ARBERR to initiate MABORT.

MCMD (110XX) An MCMD error occurs when an illegal M-bus command is issued or when bad MCPARITY is detected.

MCMD PARITY (11001) - During MWP4, MWP5, or MWP6, while in this diagnostic mode, the MCMD parity bit from the M-bus will be inverted on the memory module and the memory will issue an MABORT and assert MCPARITY error. See Section 9.4.5 for information about what cycles the memory checks for CMD parity.

MCMD INVALID (11011) - When this test mode is selected, the memory will view a memory space WRITET transaction as an unassigned MCMD value. The memory will assert INVALID_MCMD and issue MABORT.

MSTATUS (11101) An MSTATUS error occurs when an invalid parity is detected on the MSTATUS lines.

MSTATUS PARITY (11101) - In this diagnostic mode, during P7-P10 of memory reads, The MSTATUS parity bit from the M-bus will be inverted on the memory module and the memory will issue an MABORT and flag MSTATUS parity. See Section 9.4.5 for information about cycles in which the memory checks for STATUS parity.

FMDCSR<22>INHIBIT_SBE_REPORT (RW) INHIBIT SBE Reporting

This bit will prevent the memory from reporting corrected single bit errors as "CORRECTED" on the M-bus STATUS lines. Rather, SBEs will be reported as "GOOD" on the M-bus STATUS lines if they occur when this bit is asserted.

FMDCSR<23>INH_SBE_MSECTERR_LOG (RW) Inhibit Single Bit Error Logging in the MSECTERR Register during self test

This bit will prevent the memory from reporting single bit errors in the MSECTERR register. While enabled, the memory will continue to report double and multiple bit errors in MSECTERR. This bit is intended to be used if too many sections of memory are marked as "bad" in the MSECTERR register and single bit errors are suspected as the culprits.

FMDCSR<29:24>Reserved (R)

Bits 29 through 24 will be read as zero, and writing a one to these bits will have no effect.

FMDCSR<30>MODULE_ON_LINE (RW)

Bit 30 is controlled by the MEMSPEN bit in the BASEADDR register as well as the array's self test logic. When enabled, the MOL bit signifies that the array has been enabled for memory space transactions and is available. During self test, the module is not available, and the MOL bit is negated by the array.

FMDCSR<31>ERROR_FLAG_SUMMARY (RW)

This bit indicates that a memory error has occurred and that information about the error is in one or more of the following registers:

BUSCSR	BUSCTL	BUSADR
BUSDAT	ECCSYND0	ECCSYND1
ECCADDR0	ECCADDR1	MSECTERR

The error types that will set this flag are listed below:

SLAVE TIMEOUT	SLAVE ARB ERROR
ROW or COLUMN ADDRESS PARITY ERROR	SINGLE BIT ERROR
MDAL PARITY	MULTIPLE BIT ERROR
MCMD PARITY	
MSTATUS PARITY	
MCMD INVALID	

9.7.3.7. BASEADDR Register

The BASEADDR Register is a 12-bit read/write register that holds the starting address of the memory address space supported by this memory module and an enable bit which allows it to respond to memory space transactions. The starting address is programmable on 1 megabyte boundaries. Figure 9-9 shows the bit definitions for the BASEADDR Register.

```
             Name: BASEADDR     Address: XXFFFFE4#16        Access: RW

          3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
          1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
         +-+-------------------+---------------------------------------+
         |M|    STARTADDR      |              MBZ                      |
         +^+------^------------+---------------------------------------+
BASEADDR          |
          |       |
MEMSPEN+          |
                  |
STARTADDR-------+
```

Figure 9-9: BASEADDR Register

BASEADDR<31>MEMSPEN (RW) Memory Space Enable

This bit, when set, enables the memory module to respond to memory space transactions. Prior to setting this bit, the memory module will only respond to I/O space transactions.

BASEADDR<30:20>STARTADDR (RW) Base Address of memory space

The STARTADDR field is an 11-bit read/write bit field that holds the starting address, in the M-bus address map, of the memory located on this module. A read to this register returns the current STARTADDR contents. A write to this bit-field will overwrite the current contents of the STARTADDR field. On system reset, this register is cleared to 0's.

9.7.3.8. ECCADDR0 Register

The ECCADDR0 Register is an 23-bit read only register that holds the DRAM address of the the highest priority ECC error detected by the memory module for quadword 0. This register, in conjunction with the ECCSYND0 register, gives a full description of the address and data bit in error for quadword 0. Figure 9-10 shows the bit definitions for the ECCADDR0 Register.

```
               Name: ECCADDR0     Address: XXFFFFE0#16        Access: R

          3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
          1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
         +---------+-------------------------------------------+-------+
         | MBZ     |               RAMERRADDR0                 | MBZ   |
         +---------+-----------------^-------------------------+-------+
     ECCADDR0                        |
                                     |
     RAMERRADDR0 --------------------+
```

Figure 9-10: ECCADDR0 Register

ECCADDR0<26:4>RAMERRADDR0 (R) ECC Error Address (quadword 0)

This field captures the normalized memory address of the highest priority ECC error which occurred to quadword 0. Its contents can be used to help identify which DRAM has produced a failure. The contents of this register, when added to the STARTADDR field, equals the M-bus address of the transaction in which the error occurred.

9.7.3.9. ECCADDR1 Register

The ECCADDR1 Register is an 23-bit read only register that holds the DRAM address of the the highest priority ECC error detected by the memory module for quadword 1. This register, in conjunction with the ECCSYND1 register, gives a full description of the address and data bit in error for quadword 1. Figure 9-11 shows the bit definitions for the ECCADDR1 Register.

```
               Name: ECCADDR1     Address: XXFFFFDC#16        Access: R

          3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
          1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
         +---------+-------------------------------------------+-------+
         | MBZ     |               RAMERRADDR1                 | MBZ   |
         +---------+-----------------^-------------------------+-------+
     ECCADDR1                        |
                                     |
     RAMERRADDR1 --------------------+
```

Figure 9-11: ECCADDR1 Register

ECCADDR1<26:4>RAMERRADDR1 (R) ECC Error Address (quadword 1)

This field captures the normalized memory address of the highest priority ECC error which occurred to quadword 1. Its contents can be used to help identify which DRAM has produced a failure. The contents of this register, when added to the STARTADDR field, equals the M-bus address of the transaction in which the error occurred.

9.7.3.10. ECCSYND0 Register

The ECCSYND0 register is a 32 bit read/write register that records ECC error information regarding quad-word 0. Syndrome bits are saved for the highest priority error, and flags indicate the occurrence of SBE's, MBE's or both. There are also two bytes which will be used to access the check bit fields directly. These fields will be used for testing the ECC generation/checking circuits. This register, in conjunction with the ECCADDR0 register, gives a full description of the class and address of ECC error. Figure 9-12 shows the bit definitions for the ECCSYND0 Register.

```
              Name: ECCSYND0      Address: XXFFFFD8#16         Access: RW

          3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
          1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
         +---------------+---------------+-----+-----+-+-+---------------+
         |   READCBO     |     SUBCBO    | RES | ERR |M|S|     SYNDO     |
         +------^--------+-------^-------+--^--+--^--+^+^+-------^-------+
ECCSYNDO        |                |          |     |   | |        |
                |                |          |     |   | |        |
READCBO  -------+                |          |     |   | |        |
SUBCBO   ------------------------+          |     |   | |        |
RESERVED -----------------------------------+     |   | |        |
ERROVFLO -----------------------------------------+   | |        |
MBEO     ---------------------------------------------+ |        |
SBEO     -----------------------------------------------+        |
SYNDO    --------------------------------------------------------+
```

Figure 9-12: ECCSYND0 Register

ECCSYND0<7:0>SYND0 (R/W) Saved Quadword 0 ECC Syndrome Bits

The SYND0 field captures the error syndrome result for the first occurrence of the highest priority error (MBE or SBE) for Quadword 0. If only single bit errors have occurred, then only the first single bit error syndrome will be captured. i.e If any additional single bit errors occurred, the SYND0 field will not be updated and the additional syndrome data will be lost. If any multiple bit error has occurred, then the first MBE syndrome will be captured regardless of the occurrence of any previous or subsequent SBE. Any subsequent MBE will not cause this field to be updated. This field is cleared by system reset.

ECCSYND0<8>SBE0 (R/W) Single Bit Error (quadword 0)

The SBE0 bit, when set, indicates the occurrence of single bit error(s) in quadword zero since the last time the ECCSYND0 register was cleared. This bit is cleared by system reset.

ECCSYND0<9>MBE0 (R/W) Multiple Bit Error (quadword 0)

The MBE0 bit, when set, indicates the occurrence of multiple bit error(s) in quadword zero since the last time the ECCSYND0 register was cleared. This bit is cleared by system reset.

ECCSYND0<12:10>ERROVFL0 (R/W) Error Count (quadword 0)

The ERROVFL0 field indicates the sum -1 of MBE and SBE errors since the last time that the ECCSYND0 register was cleared. i.e. If one error has occurred, the ERROVFL0 field will be zero; and the error information will represent the last error that occurred to this quadword. If five errors have occurred, then this field will be 4 (100 #2), indicating that the information for four errors has been lost. The ERROVFL0 field will never overflow. The number 7 (111 #2) indicates that at least eight errors have occurred in quadword zero since the last time the ECCSYND0 register was cleared. This field is cleared by system reset.

ECCSYND0<15:13>RESERVED (R0)

This field is reserved for future definition. It will always read back as zero.

ECCSYND0<23:16>SUBCB0 (R/W) Substitute Check Bits (quadword 0)

The SUBCB0 field is used during diagnostic check of the ECC function. When the ECC diag mode 10#2 is set in the FMDCSR, subsequent memory space reads will cause the contents of this field to be used as the read check bits for quadword 0 instead of the actual check bits read from the DRAM. This register can be used to verify that the ECC circuitry can flag all errors and correct single bit errors. This field is cleared by system reset.

ECCSYND0<31:24>READCB0 (R/W) Read Check Bits (quadword 0)

The READCB0 field serves two purposes.

When ECC diagnostic mode 01#2 is set in the FMDCSR, The READCB0 field is used to access the quadword 0 check bit field on memory reads. During every memory read transaction, the contents of the quadword 0 checkbit RAMs will be loaded into the READCB0 field.

When ECC diagnostic mode 11#2 is set in the FMDCSR, a memory write transaction will cause the ECC generated write check bits for quadword 0 to be loaded into the READCB0 field. This field will be cleared by system reset.

9.7.3.11. ECCSYND1 Register

The ECCSYND1 register is a 32 bit read/write register that records ECC error information regarding quadword 1. Syndrome bits are saved for the highest priority error, and flags indicate the occurrence of SBE's, MBE's or both. There are also two bytes which will be used to access the check bit fields directly. These fields will be used for testing the ECC generation/checking circuits. This register, in conjunction with the ECCADDR1 register, gives a full description of the class and address of ECC error. Figure 9-13 shows the bit definitions for the ECCSYND1 Register.

```
           Name: ECCSYND1     Address: XXFFFFD4#16       Access: RW

       3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
       1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
      +---------------+---------------+-----+-----+-+-+---------------+
      |    READCB1    |     SUBCB1    | RES | ERR |M|S|     SYND1     |
      +-------^-------+-------^-------+--^--+--^--+^+^+-------^-------+
ECCSYND1     |                |          |     |   | |        |
             |                |          |     |   | |        |
READCB1 -----+                |          |     |   | |        |
SUBCB1 -----------------------+          |     |   | |        |
RESERVED --------------------------------+     |   | |        |
ERROVFL1 --------------------------------------+   | |        |
MBE1 ----------------------------------------------+ |        |
SBE1 ------------------------------------------------+        |
SYND1 --------------------------------------------------------+
```

Figure 9-13: ECCSYND1 Register

ECCSYND1<7:0>SYND1 (R/W) Saved Quadword 1 ECC Syndrome Bits

The SYND1 field captures the error syndrome result for the first occurrence of the highest priority error (MBE or SBE) for Quadword 1. If only single bit errors have occurred, then only the first single bit error syndrome will be captured. i.e If any additional single bit errors occurred, the SYND1 field will not be updated and the additional syndrome data will be lost. If any multiple bit error has occurred, then the first MBE syndrome will be captured regardless of the occurrence of any previous or subsequent SBE. Any subsequent MBE will not cause this field to be updated.

ECCSYND1<8>SBE1 (R/W) Single Bit Error (quadword 1)

The SBE1 bit, when set, indicates the occurrence of single bit error(s) in quadword one since the last time the ECCSYND1 register was cleared.

ECCSYND1<9>MBE1 (R/W) Multiple Bit Error (quadword 1)

The MBE1 bit, when set, indicates the occurrence of multiple bit error(s) in quadword one since the last time the ECCSYND1 register was cleared.

ECCSYND1<12:10>ERROVFL1 (R/W) Error Count (quadword 1)

The ERROVFL1 field indicates the sum -1 of MBE and SBE errors since the last time that the ECCSYND1 register was cleared. i.e. If one error has occurred, the ERROVFL1 field will be zero; and the error information will represent the last error that occurred to this quadword. If five errors have occurred, then this field will be 4 (100 #2), indicating that the information for four errors has been lost. The ERROVFL1 field will never overflow. The number 7 (111 #2) indicates that at least eight errors have occurred in quadword one since the last time the ECCSYND1 register was cleared.

ECCSYND1<15:13>RESERVED (R0)

This field is reserved for future definition. It will always read back as zero.

ECCSYND1<23:16>SUBCB1 (R/W) Substitute Check Bits (quadword 1)

The SUBCB1 field is used during diagnostic check of the ECC function. When the ECC diag mode 10#2 is set in the FMDCSR, subsequent memory space reads will cause the contents of this field to be used as the read check bits for quadword 1 instead of the actual check bits read from the DRAM. This register can be used to verify that the ECC circuitry can flag all errors and correct single bit errors. This field is cleared by system reset.

ECCSYND1<31:24>READCB0 (R/W) Read Check Bits (quadword 1)

The READCB0 field serves two purposes.

When ECC diagnostic mode 01#2 is set in the FMDCSR, The READCB0 field is used to access the quadword 0 check bit field on memory reads. During every memory read transaction, the contents of the quadword 1 checkbit RAMs will be loaded into the READCB0 field.

When ECC diagnostic mode 11#2 is set in the FMDCSR, a memory write transaction will cause the ECC generated write check bits for quadword 1 to be loaded into the READCB0 field. This field will be cleared by system reset.

9.7.3.12. MSECTERR Register

The MSECTERR Register is an 32-bit read only register. It holds an indication of what section(s) of this module's memory address space had one or more errors during the last DRAM array self test run on this memory module. Figure 9-14 shows the bit definitions for the MSECTERR Register.

```
                  Name: MSECTERR      Address: XXFFFFD0#16         Access: R

        3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
        1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
       +---------------------------------------------------------------+
       |                            ERRSECTION                         |
       +-----------------------------^---------------------------------+
MSECTERR                             |
          !                          |
ERRSECTION---------------------------+
```

Figure 9-14: MSECTERR Register

MSECTERR<31:0>ERRSECTION (R) Saved Memory Section With Detected Errors

Each bit in this bit field represents one thirty-second of the memory address space present on this module. During the DRAM array self test, any errors encountered [SBEs(if FMDCSR<23>Ignore SBEs is cleared) or MBEs] will set the ERRSECTION bit appropriate to the address of that error. For example, on a 16M Byte module, 1/32 of 16M Bytes is 512K Bytes. If an error(s) occurred during testing an address from 0 to 512K the MEMSECTERR<0> would be set. If an error(s) occurred from 513K to 1M, then MSECTERR<1> would be set. Both bits would be set if one or more errors occurred in each section. In the case of a 32M Byte module, each bit of the MSECTERR register would represent 1/32 of that address space, or 1M Byte.

The contents of this register, after the DRAM array self test has been run, can be used as an index into those sections of the module that had error(s) detected. For further details on which 512 byte pages are in error, the entire section must be scanned looking for the quadwords that have been intentionally marked by the test with MBEs. This facility can be used to speed up the process of building a "Bad Page Table" for the operatin system especially if only a small section of the module has errors.

This register is cleared at the start of self test and on system reset. Writing to this register has no effect.

9.7.3.13. MBUSSIG Register

The MBUSSIG Register is an 12-bit read/write register that holds the signature of key control nodes within the FMDC that are associated with the M-bus interface state machine. Figure 9-15 shows the bit definitions for the MBUSSIG Register.

```
              Name: MBUSSIG      Address: XXFFFFCC#16        Access: RW

           3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
           1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
          +----------------------------------------+-----------------------+
          |                     MBZ                |        MRESIDUE       |
          +----------------------------------------+-----------^-----------+
      MBUSSIG                                                  |
                                                               |
      MRESIDUE-------------------------------------------------+
```

Figure 9-15: MBUSSIG Register

MBUSSIG<11:0>MRESIDUE(RW) M-bus control state machine LFSR residue.

> This 12 bit field does not have a set of bit definitions in the classic sense. Rather, it contains a value (just a number) which is the residue of the control nodes fed into a LFSR as a result of one or more M-bus transactions executed by the FMDC and is a function of any initial value or previous residue. This register can be set to an initial value (nonzero if desired) via an I/O Write transaction. It can be read via an I/O Read transaction and will contain the signature of the transactions that have transpired since the last I/O Write to this register or since the last I/O Read transaction. The reason for this register is that a sequence of M-bus transactions can be found (via simulation/fault grading) which gives a nearly complete indication as to whether this state machine is functioning properly or not. The actual test sequences and corresponding MBUSSIG<11:0> values will be published in another document at a later date.

9.7.3.14. DRAMSIG Register

The DRAMSIG Register is an 12-bit read/write register that holds the signature of key control nodes within the FMDC that are associated with the DRAM interface state machine. Figure 9-16 shows the bit definitions for the DRAMSIG Register.

```
              Name: DRAMSIG      Address: XXFFFFC8#16        Access: RW

           3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
           1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
          +----------------------------------------+-----------------------+
          |                     MBZ                |        DRESIDUE       |
          +----------------------------------------+-----------^-----------+
      DRAMSIG                                                  |
                                                               |
      DRESIDUE-------------------------------------------------+
```

Figure 9-16: DRAMSIG Register

DRAMSIG<11:0>DRESIDUE(RW) DRAM control state machine LFSR residue

This 12 bit field does not have a set of bit definitions in the classic sense. Rather, it contains a value (just a number) which is the residue of the control nodes fed into a LFSR as a result of one or more transactions that result in a cycling of the DRAM state machine; (ie: M-bus or self test Octaword Writes, M-bus or selftest Octaword Reads, internal or external Refresh). The residue value is a function of any initial value or previous residue. This register can be set to an initial value (nonzero if desired) via an I/O Write transaction. It can be read via an I/O Read transaction and will contain the signature of the transactions that have transpired since the last I/O Write to this register or since the last I/O Read transaction. The reason for this register is that a sequence of M-bus transactions can be found (via simulation/fault grading) which gives a nearly complete indication as to whether the DRAM state machine is functioning properly or not. The actual test sequences and corresponding DRAMSIG<11:0> values will be published in another document at a later date.

9.7.3.15. SELFSIG Register

The SELFSIG Register is an 12-bit read/write register that holds the signature of key control nodes within the FMDC that are associated with the DRAM array self test state machine. Figure 9-17 shows the bit definitions for the SELFSIG Register.

```
        Name: SELFSIG       Address: XXFFFFC4#16        Access: RW

       3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
       1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
      +---------------------------------------+-----------------------+
      !                   MBZ                 !        SRESIDUE       !
      +---------------------------------------+-----------^-----------+
SELFSIG                                                   |
                                                          !
SRESIDUE--------------------------------------------------+
```

Figure 9-17: SELFSIG Register

SELFSIG<11:0>SRESIDUE (RW) DRAM array self test control state machine LFSR residue.

This 12 bit field does not have a set of bit definitions in the classic sense. Rather, it contains a value (just a number) which is the residue of the control nodes fed into a LFSR as a result of cycles executed by performing the DRAM array self test. The residue value is a function of any initial value or previous residue. This register can be set to an initial value (nonzero if desired) via an I/O Write transaction. It can be read via an I/O Read transaction and will contain the signature of the sequences performed by the self test state machine since the last I/O Write to this register or since the last I/O Read transaction.

The reason for this register is that a a good DRAM self test sequence will have only one valid signature value for a successful test completion. One can verify that the test was performed successfully by examining this register for the correct value at the tests end.

The value that should be found in SELFSIG<11:0> after DRAM array self test has been performed will be derived empirically. The simulation/ fault grading) option used on the other LFSR's is not an option due to the large number of cycles performed by the DRAM array self test and the slowness of simulation. The actual SELFSIG<11:0> values will be published in another document at a later date.

9.7.3.16. LEDLATCH Register

The LEDLATCH register is a 6-bit register that stores the current state of the diagnostic/self-test LEDs. It is a read/write register that is accessible through I/O space to every processor in the Firefox system. Figure 9-18 shows the bit definitions for the LEDLATCH Register.

```
          Name: LEDLATCH      Address: XXFFFFC0#16         Access: RW

       3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
       1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
      +-----------------------------------------------------------------+
      |                       MBZ                             | LEDVALUES |
      +-------------------------------------------------------------^------+
 LEDLATCH                                                     |
                                                              |
 LEDVALUES----------------------------------------------------+
```

Figure 9-18: LEDLATCH Register

LEDLATCH<5:0>LEDVALUES (RW) External Self-test LED Values

Self-test code uses the LEDVALUES bit-field to illuminate the LEDs on the memory module. A read to this register returns the contents of the on-chip FMDC register which is a shadow of the externally implemented LEDLATCH. A write to this bit-field will write the corresponding bits in the external and internal latch, as well as light the associated LEDs. For each bit in this bit field, a 0 will illuminate an LED on the module handle and a 1 will turn it off. On system reset, this register is cleared to 0's, resulting in all of the LEDs being illuminated.

9.7.3.17. EXP_MBUSSIG Register

The EXP_MBUSSIG register is a 12 bit read only register that holds the expect value of the MBUS signature LFSR's. Figure 9-19 shows the bit definitions for the EXP_MBUSSIG register.

```
         Name: EXP_MBUSSIG      Address: XXFFFFBC#16       Access: R

       3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
       1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
      +----------------------------------------+------------------------+
      |                  MBZ                   |        MEXPECT         |
      +----------------------------------------+-----------^------------+
 EXP_MBUSSIG                                               |
                                                           |
 MEXPECT --------------------------------------------------+
```

Figure 9-19: EXP_MBUSSIG Register

EXP_MBUSSIG<11:0>MEXPECT (R) M-bus state machine signature expect register

The value read from the MEXPECT field is equal to the the expected value to be read from the MBUSSIG register after a predetermined set of transactions have been run on the M-bus. The set of transactions to be run is **TBD**.

9.7.3.18. EXP_DRAMSIG Register

The EXP_DRAMSIG register is a 12 bit read only register that holds the expect value of the DRAM state machine signature LFSR's. Figure 9-20 shows the bit definitions for the EXP_DRAMSIG register.

```
              Name: EXP_DRAMSIG      Address: XXFFFFB8#16      Access: R

          3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
          1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
         +---------------------------------------+-----------------------+
         |                  MBZ                  |        DEXPECT        |
         +---------------------------------------+-----------^-----------+
    EXP_DRAMSIG                                              |
                                                             |
    DEXPECT -------------------------------------------------+
```

Figure 9-20: EXP_DRAMSIG Register

EXP_DRAMSIG<11:0>DEXPECT (R) DRAM state machine signature expect register.

The value read from the DEXPECT field is equal to the the expected value to be read from the DRAMSIG register after a predetermined set of DRAM transactions have been run on the memory under test. The set of transactions to be run is **TBD**.

9.7.3.19. EXP_SELFSIG Register

The EXP_SELFSIG register is a 12 bit read only register that holds the expect value of the Self Test state machine signature LFSR's. Figure 9-21 shows the bit definitions for the EXP_SELFSIG register.

```
              Name: EXP_SELFSIG      Address: XXFFFFB4#16      Access: R

          3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
          1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
         +---------------------------------------+-----------------------+
         |                  MBZ                  |        SEXPECT        |
         +---------------------------------------+-----------^-----------+
    EXP_SELFSIG                                              |
                                                             |
    SEXPECT -------------------------------------------------+
```

Figure 9-21: EXP_SELFSIG Register

EXP_SELFSIG<11:0>SEXPECT (R) Self Test state machine signature expect register.

The value read from the SEXPECT field is equal to the the expected value to be read from the SELFSIG register after completion of the self test function.

9.8. ECC Strategy and Goals

The Firefox memory array will be organized as a 64 bit wide data path with an associated 8 bit ECC code. We will use a modified Hamming code whose elements will be selected such that they satisfy the following goals:

- Associate an odd # of data bits with each check bit. This will facilitate self test by allowing complemented data to generate complemented check bits. This will alleviate a separate pass through the memory address space during self test just to verify that every check bit can hold a "0" or a "1".
- Use all the codes of 3 - "1's" as valid codes and supplement these with specific 5 -"1's" codes. The 5 -"1's" codes used will have a "1" in the check bit position that is not complemented in the "WRITE KNOWN BAD DATA" operation (ie: C<7>).
- Assure that, when we generate check bits and invert check bits C<6:0> for the "WRITE KNOWN BAD DATA" indication, no valid SBE code is selected.
- Include the address bit which maps to the low order "BANK ADDRESS" along with the addresses which map to the "ROW ADDRESS" in a parity calculation which maps to its own error syndrome (ie: ROW PARITY error). This error will be mapped to an uncorrectable error.
- Include the address bit which maps to the high order "BANK ADDRESS" along with the addresses which map to the "COLUMN ADDRESS" in a parity calculation which maps to its own error syndrome (ie: COLUMN PARITY error). This error will be mapped to an uncorrectable error.
- Use the CALYPSO memory system scheme of generating XOR trees for groups of 6 data bits. However, make sure that those data bits that see 2 AC loads are distributed such that no one see's an exceptionally high AC load.
- Select either EVEN or ODD parity for each check bit so that the common multiple bit failure of all zeros or all ones (including check bits) is detectable as an uncorrectable error. This must be true for all combinations of row and column address parity.

9.8.1. Single Bit Error (SBE) Codes

	SY7	SY6	SY5	SY4	SY3	SY2	SY1	SY0	"Syndrome"(Hex)
No Err	0	0	0	0	0	0	0	0	00
Codes with Three Data Contributors									
D<0>	0	0	0	0	0	1	1	1	07
D<1>	0	0	0	0	1	0	1	1	0B
D<2>	0	0	0	0	1	1	0	1	0D
D<3>	0	0	0	0	1	1	1	0	0E
D<4>	0	0	0	1	0	0	1	1	13
D<5>	0	0	0	1	0	1	0	1	15
D<6>	0	0	0	1	0	1	1	0	16
D<7>	0	0	0	1	1	0	0	1	19
D<8>	0	0	0	1	1	0	1	0	1A
D<9>	0	0	0	1	1	1	0	0	1C
D<10>	0	0	1	0	0	0	1	1	23
D<11>	0	0	1	0	0	1	0	1	25
D<12>	0	0	1	0	0	1	1	0	26
D<13>	0	0	1	0	1	0	0	1	29
D<14>	0	0	1	0	1	0	1	0	2A
D<15>	0	0	1	0	1	1	0	0	2C
D<16>	0	0	1	1	0	0	0	1	31
D<17>	0	0	1	1	0	0	1	0	32
D<18>	0	0	1	1	0	1	0	0	34
D<19>	0	0	1	1	1	0	0	0	38

	SY7	SY6	SY5	SY4	SY3	SY2	SY1	SY0	"Syndrome"(Hex)
D<20>	0	1	0	0	0	0	1	1	43
D<21>	0	1	0	0	0	1	0	1	45
D<22>	0	1	0	0	0	1	1	0	46
D<23>	0	1	0	0	1	0	0	1	49
D<24>	0	1	0	0	1	0	1	0	4A
D<25>	0	1	0	0	1	1	0	0	4C
D<26>	0	1	0	1	0	0	0	1	51
D<27>	0	1	0	1	0	0	1	0	52
D<28>	0	1	0	1	0	1	0	0	54
D<29>	0	1	0	1	1	0	0	0	58
D<30>	0	1	1	0	0	0	0	1	61
D<31>	0	1	1	0	0	0	1	0	62
D<32>	0	1	1	0	0	1	0	0	64
D<33>	0	1	1	0	1	0	0	0	68
D<34>	0	1	1	1	0	0	0	0	70
D<35>	1	0	0	0	0	0	1	1	83
D<36>	1	0	0	0	0	1	0	1	85
D<37>	1	0	0	0	0	1	1	0	86
D<38>	1	0	0	0	1	0	0	1	89
D<39>	1	0	0	0	1	0	1	0	8A
D<40>	1	0	0	0	1	1	0	0	8C
D<41>	1	0	0	1	0	0	0	1	91
D<42>	1	0	0	1	0	0	1	0	92
D<43>	1	0	0	1	0	1	0	0	94
D<44>	1	0	0	1	1	0	0	0	98
D<45>	1	0	1	0	0	0	0	1	A1
D<46>	1	0	1	0	0	0	1	0	A2
D<47>	1	0	1	0	0	1	0	0	A4
D<48>	1	0	1	0	1	0	0	0	A8
D<49>	1	0	1	1	0	0	0	0	B0
D<50>	1	1	0	0	0	0	0	1	C1
D<51>	1	1	0	0	0	0	1	0	C2
D<52>	1	1	0	0	0	1	0	0	C4
D<53>	1	1	0	0	1	0	0	0	C8
D<54>	1	1	0	1	0	0	0	0	D0
D<55>	1	1	1	0	0	0	0	0	E0

Codes with Five Data Contributors

	SY7	SY6	SY5	SY4	SY3	SY2	SY1	SY0	"Syndrome"(Hex)
D<56>	1	0	0	1	0	1	1	1	97
D<57>	1	0	0	1	1	0	1	1	9B
D<58>	1	0	0	1	1	1	0	1	9D
D<59>	1	0	0	1	1	1	1	0	9E
D<60>	1	1	1	0	0	1	0	1	E5
D<61>	1	1	1	0	0	1	1	0	E6
D<62>	1	1	1	1	0	1	0	0	F4
D<63>	1	1	1	1	1	0	0	0	F8

	SY7	SY6	SY5	SY4	SY3	SY2	SY1	SY0	"Syndrome"(Hex)
Codes with One Data Contributor									
CB0	0	0	0	0	0	0	0	1	01
CB1	0	0	0	0	0	0	1	0	02
CB2	0	0	0	0	0	1	0	0	04
CB3	0	0	0	0	1	0	0	0	08
CB4	0	0	0	1	0	0	0	0	10
CB5	0	0	1	0	0	0	0	0	20
CB6	0	1	0	0	0	0	0	0	40
CB7	1	0	0	0	0	0	0	0	80

9.8.2. Multiple Bit Error Syndromes

Included here are the most probable causes for specific codes.

9.8.2.1. Multiple Bit Error (MBE) Syndromes with a Singular Cause

SY7	SY6	SY5	SY4	SY3	SY2	SY1	SY0	"Syndrome" (Hex)
0	1	1	1	1	1	1	1	7F "Written known bad data"
1	1	1	0	1	1	0	0	EC "Row address parity error"
1	1	0	1	1	1	0	0	DC "Column address parity Error"

9.8.2.2. Multiple Bit Error Syndromes Due to Catastrophic Failure

All "0"s or all "1"s in both data and check bits.

SY7	SY6	SY5	SY4	SY3	SY2	SY1	SY0	"Syndrome"(Hex)
0	0	0	0	0	0	1	1	03 Even row/even column
1	1	1	0	1	1	1	1	EF Odd row/even column
1	1	0	1	1	1	1	1	DF Even row/odd column
0	0	1	1	0	0	1	1	33 Odd row/odd column

9.8.3. Check Bit Implementation

CB0 and CB1 are calculated for ODD parity (an odd number of "1"s counting the CB).

CB2, CB3, CB4, CB5, CB6, CB7 are calculated for EVEN parity (an even number of "1"s counting the CB).

9.9. Module Self Test and Cooperative Test Strategy

In this section I will describe the features provided by this module to perform a self test on its DRAM array, and, those features that can be used, in concert with a processor, to verify the integrity of this memory unit.

9.9.1. Overview

Testing of any electronic subsystem can be broken down into testing of its component subblocks and, subsequently, testing of the interactions between these blocks.

The approach we have taken with this memory module is to divide the overall test task into component pieces that are closely aligned with the subblocks of the high level block diagram. Each major subblock in the high level block diagram is tested using one of the two following strategies:

- **Stimulus/Response.** For some of the subblocks, we are using a stimulus/response technique where we present (or are presented with) a sequence of test vectors and check for the appropriate response.
- **Signature Analysis.** For other subblocks, we are presented with sequences of operations which we execute. While we are executing these operations, we are collecting information from critical nodes internal to the module in a Linear Feedback Shift Register (LFSR). This LFSR can be examined via an I/O Read to a designated register address and its contents can be compared with the expected residue from this sequence (determined via simulation).

9.9.2. Address Path Testing

This block of logic will be tested via the Stimulus/Response technique. The details of this testing are **TBD**.

9.9.3. Data Path Testing

This block of logic will be tested via the Stimulus/Response technique. The details of this testing are **TBD**.

9.9.4. M-Bus State Machine Testing

This block of logic will be tested via the Signature Analysis technique. The details of this testing are **TBD**.

9.9.5. DRAM State Machine Testing

This block of logic will be tested via the Signature Analysis technique. The details of this testing are **TBD**.

9.9.6. DRAM Array Self Test

This block of logic will be tested via the Stimulus/Response technique. The details of this testing are given in the following sec- tions.

9.9.6.1. DRAM Array Self Test Sequence

The purpose of the DRAM self test is to verify that each DRAM device on the memory module is capable of storing both a "0" and a "1". Also, we would like to gain an additional measure of confidence in the DRAM array and its associated drivers by addressing the memory in a pseudorandom fashion and storing pseudorandom data. This style of addressing and data storage is effective in uncovering some addressing mode failures and some data bit dependencies in a very time efficient manner.

The test sequence consists logically of three passes through the modules memory address space (these three passes are time equivalent to 3 WRITE passes and 2 READ passes). The three passes are:

1. WRITE memory with a background pattern of pseudorandom DATA.
2. READ and check memory (should be DATA) - WRITE inverse DATA.
3. READ and check memory (should be inverse DATA)- WRITE all 0's DATA to initialize memory).

The normal ECC circuitry is used to check the READ DATA for accuracy. Also, the ECC code used was selected such that "inverse DATA" gives you "inverse" check bits. This allows us to verify that every DRAM device stores a "0" and a "1" in only two WRITE/READ passes. Also, two special LFSRs (Linear Feedback Shift Registers) are used; one to generate the pseudorandom ADDRESS and one to generate pseudorandom DATA. The polynomials selected for both of these LFSRs are prime (ie: they repeat only after n cycles where n = length of LFSR).

9.9.6.1.1. DRAM Array Test Flow

A. Perform Octaword WRITEs throughout the entire modules memory space using:
   a. LFSR generated ADDRESSES

b. LFSR generated DATA

B. Perform a pair of cycles which are logically coupled - an Octaword READ followed immediately by an Octaword WRITE (to the same ADDRESS) as follows:

During the Octaword READ Cycle

a. Use LFSR generated ADDRESSES - Hold this ADDRESS for the READ-WRITE Cycle pair.

b. Turn the ECC correction circuitry off

c. Use the ECC circuitry to check the READ DATA:

i. If READ DATA is OK:

a) Recirculate -(DATA) to be written into the WRITE DATA BUFFER.

ii. If READ DATA is in error:

a) Recirculate DATA to be written into the WRITE DATA BUFFER

b) Write the FLAG_BAD bit into the WRITE DATA BUFFER for this quadword. (This causes the ECC circuitry to generate complement check bits for CB<6:0>.)

c) Flag the appropriate MSECTERR Register bit. The actual bit location is determined by the ADDRESS and memory module size (ie 8,16,32,64, or 128MByte).

During the Octaword WRITE Cycle

a. Using the same ADDRESS as the READ cycle - Write the Octaword stored in the WRITE DATA BUFFER during the READ cycle.

C. Perform a pair of cycles which are logically coupled - an Octaword READ followed immediately by an Octaword WRITE (to the same ADDRESS) as follows:

During the Octaword READ Cycle

a. Use LFSR generated ADDRESSES - Hold this ADDRESS for the READ-WRITE Cycle pair.

b. Turn the ECC correction circuitry off

c. Use the ECC circuitry to check the READ DATA:

i. If READ DATA is OK:

a) Recirculate all 0's DATA to be written into the WRITE DATA BUFFER.

ii. If READ DATA is in error:

a) Recirculate DATA to be written into the WRITE DATA BUFFER

b) Write the FLAG_BAD bit into the WRITE DATA BUFFER for this quadword. (This causes the ECC circuitry to generate complement check bits for CB<6:0>.)

c) Flag the appropriate MSECTERR Register bit. The actual bit location is determined by the ADDRESS and memory module size (ie 8,16,32,64 or 128MByte).

During the Octaword WRITE Cycle

a. Using the same ADDRESS as the READ cycle - Write the Octaword stored in the WRITE DATA BUFFER during the READ cycle.

9.9.6.2. DRAM Self Test State Machine Verification

This block of logic will be tested via the Signature Analysis technique. The details of this testing are **TBD**.

9.9.7. Other Test Hooks

Other test hooks is a category into which falls all those logic blocks that are used by module testers to do some special isolation or data mapping to accomplish their goals. It may include special I/O register bits that modify the behavior of a specific logic block such as mapping the ECC bits into an I/O register for visibility during TCY (temperature cycling) testing. Or, it may be special Gate Array input pins, such as one that tristates all Gate Array outputs to test external interface logic and the DRAM array from an

EXCEL 407 memory tester isolating the Gate Array from other external logic).

The details of these test hooks are **TBD**.